Trabajo Fin de Grado
Ingeniería Electrónica Robótica y Mecatrónica

Control de temperatura para sistema de impresión 3D

Autor: Fidel González Leiva

Tutor: Manuel Ángel Perales Esteve

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado
Ingeniería Electrónica Robótica y Mecatrónica

Control de temperatura para sistema de impresión 3D

Autor:

Fidel González Leiva

Tutor:

Manuel Ángel Perales Esteve

Profesor Titular

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado: Control de temperatura para sistema de impresión 3D

Autor: Fidel González Leiva
Tutor: Manuel Ángel Perales Esteve

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Es imposible no reconocer que tanto este trabajo, como la carrera se la debo plenamente a mi madre, mi padre y mi hermano quienes me han apoyado tanto moral como económicamente siempre. Agradecerles tanto como me han ofrecido en la vida.

Gracias a toda la gente que me soportó durante todo el grado o parte de él y quienes siempre me han tendido su mano cuando lo necesité. Por último agradecer a todas las profesoras y prefesores que me han hecho valorar el saber y me han trasnsmitido conocimientos y ganas durante toda mi vida académica.

Fidel González Leiva

Sevilla, 2018

Índice Abreviado

Índice

Notación

p. ej.	Por ejemplo
3D	3 dimensiones
SLA	Aparato de estereolitografía
FDM	Modelado por deposición de fundido
LOM	Manufacturado de objetos mediante laminado
SLS	Sinterizado selectivo por láser
FFF	Fabricación mediante fusión de filamento
DMLS	Sinterizado directo de metal por láser
EBM	Fabricación por haz de electrones
EBF^3	Fabricación de forma libre por haz de electrones
LOM	Fabricacion de objetos mediante laminado
CAD	Diseño asistido por ordenador
RepRap	Prototipadora rápida de replicantes
FabLab	Laboratorio de fabricación
CIEMAT	Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas
UC3M	Universidad Carlos III de Madrid
ABS	Acrilonitrilo butadieno estireno
PLA	Ácido poli-láctico
PET	Tereftalato de polietileno
TPE	Elastómero termoplástico
PC	Policarbonato
LGPL	Licencia pública general reducida de GNU
IDE	Entorno de desarrollo integrado
TI	Texas Instrument
IoT	Internet de las cosas
MSP	Procesador de señales mixtas
CNC	Control numérico por computador
GPPS	Poliestireno de propósito general
CPU	Unidad central de procesamiento
RISC	Computador con conjunto de instrucciones reducidas
E/S	Entrada/Salida
PWM	Modulación por ancho de pulsos

1 Introducción

Es el momento de redefinir. Tenemos que reinventar la rueda, pero libre.

Juan Gónzalez Gómez, 2013

Hace ya medio siglo, en 1966, el escritor y científico Arthur C. Clarke profetiza una máquina a la que describe como "el final de todas las invenciones". Él llamará a esta invención "La Replicadora" la cual será capaz de replicar cualquier otro objeto tridimensional que puedas encontrar. Tan solo 20 años después, la ficción se funde con la realidad una vez más cuando Chuck Hull inventa la primera máquina que podemos concebir como una impresora 3D. Su nombre fue SLA (StertereoLithography Apparatus) y a diferencia de las impresoras que conocemos hoy día estas utilizaban un haz de luz para solidificar una resina, construyendo capa por capa el objeto deseado.

Poco después en 1989 S. Scott Crump patenta una nueva máquina con la misma funcionalidad, pero esta vez se he habla del modelado mediante depositación de material fundido FDM (Fused Deposition Modeling), técnica que años más tarde se volverá de las más extendida entre los usuarios. El crecimiento de la impresión 3D transcurre lentamente durante la década de los 90 y comienzos del nuevo milenio, con la invención de nuevas técnicas como manufacturado de objetos mediante laminado (LOM) o el siterizado selectivo por láser (SLS).

Durante esta temprana edad de las "máquinas replicadoras" cada invención iba acompañada de una patente, lo que hizo que el conocimiento de estas tecnologías fuera privado y su precio muy elevado. Pero en el año 2005 nace el proyecto RepRap cuando Adrian Bowyer crea un blog con el propósito de crear una máquina capaz de replicar la mayor parte de ella misma. Esta impresora se basa en la técnica FDM, aprovechando la espiración de la patente de Scott Crump, aunque por motivos legales se acuña el término FFF que hace referencia a la fabricación mediante la fusión de filamento. La misma idea que se desarrolla en la década de los 90, comienza así a expandirse por el mundo de manera mucho más económica, haciendo que la impresión 3D salga del mundo de la industria a la casa. La gran diferencia de la aparición de RepRap con respecto a los anteriores acontecimientos es que este proyecto es un proyecto comunitario, es decir abierto a todo el mundo, de forma que cualquier persona puede coger, mejorar o reinventar este nuevo diseño de maquinas replicantes, lo que desencadenó la aparición de nuevos modelos de máquinas autoreplicantes, así como una fuerte comunidad en la que cada día más y más gente aportan tanto la información necesaria para que cualquiera monte su nueva impresora, así como nuevas soluciones a los problemas que acarrean éstas.

1.1 Tecnologías de impresión 3D

Al referirnos a las tecnologías impresión 3D realmente estamos haciendo alusión a las tecnologías de fabricación aditiva, es decir, un proceso de agregación de materia hasta obtener el producto deseado. Este nuevo concepto rompe con los esquemas de fabricación hasta entonces utilizados como las fresadoras (fabricación sustractiva) que a partir de una producto inicial (p. ej. listón de madera o pieza de metal) se erosiona gradualmente hasta obtener el producto final. Dentro de esta fabricación aditiva se encuentran múltiples técnicas en función del material que se desea tratar (polímeros, poliácidos, metales, etc.), del

entorno en el que se va a fabricar (superficie terrestre, gravedad 0) y otros múltiples factores, como las restricciones del mercado.

Todas las técnicas de fabricación aditivas tienen en común la necesidad de un ordenador, por un lado para el diseño del modelo del objeto haciendo uso de herramientas CAD; y por otra parte, como elemento de control de la maquinaria correspondiente, enviando las ordenes necesarias para la debida actuación en función del objeto que se quiera obtener.

Por otra parte, pese a que todas las técnicas se basan en la adición por capas del material deseado, el cómo se añadan éstas o los materiales de las que se parten son muy diferentes. A continuación se presentan las tecnologías más frecuentes en la actualidad.

1.1.1 Basadas en láser

Varias de las técnicas que se frecuentan hoy día tienen un principio de funcionamiento bastante similar aunque difieren en ciertos factores tales como el material empleado para la impresión o el principio físico desencadenante de la solidificación de dicho material, pero que serán englobados bajo un mismo apartado debido a su parecido.

Estereolitografía (SLA)

Como se mencionó anteriormente, ésta es la primera de las tecnologías de impresión 3D de la historia. Su acrónimo proviene del inglés StereoLithography Apparatus, y se basa en la fotopolimerización. Éste es el proceso de juntar varios monómeros (moléculas de pequeña masa molecular) con otros mediante radiación ultravioleta formando un polímero, el cual se irá formando capa a capa hasta obtener el producto deseado. Para realizar esto, se dispone de un depósito el cual se rellenará de resina de fotopolímero (1), en cuyo fondo tendremos una plataforma levadiza (2) que irá variando su cota capa a capa. A su vez, un láser, o en su defecto un haz de luz con componentes ultravioleta, es intensificado mediante un conjunto de lentes y será reflejado en un espejo con dos grados de libertad (3). Con dicho espejo se irá realizando un barrido a cada capa solidificando dicha resina hasta obtener el objeto deseado.

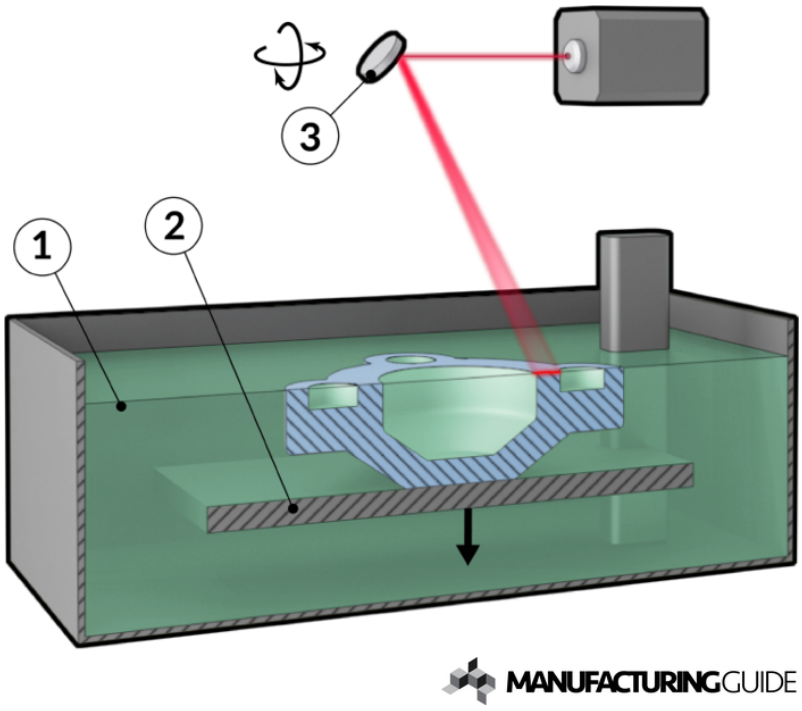


Figura 1.1 Esquema de funcionamiento de una máquina SLA.

Una de las principales ventajas del SLA es la calidad del acabado de las piezas que genera, así como las características mecánicas del material usado. Por otra parte esta es una tecnología bastante cara ya que llega

desde unos 400€ (pocos común) hasta fácilmente centenas de miles de euros. Además esta tecnología a menudo requiere tratamientos posteriores a la impresión debido a que el material es vulnerable a factores como la humedad o la exposición a la luz natural, a lo que se le suma que el volumen de impresión de estas máquinas suele ser bastante limitado.

Sinterizado Selectivo por Láser (SLS) y Sinterizado de Metal Directo por Láser (DMLS)

El sinterizado es una técnica que se basa en la producción de piezas sólidas mediante el calentamiento, sin llegar a la temperatura de fusión, de conglomerados de polvo. Tanto el SLS como el DMLS parten de de 2 contenedores (o 3 en función del modelo) en cuyo fondo hay un pistón móvil. Uno de éstos contenedores comenzará con el pistón elevado, que será la zona en la que se forme el objeto, y los demás contenedores con el pistón en la zona inferior ya que estarán llenos del polvo que se vaya a tratar. A cada capa, el pistón de la zona de impresión irá bajando a la vez que el resto sube de forma que un tambor nivelador esparcirá homogéneamente la capa de polvo de un contenedor a otro. Una vez que la zona de impresión tiene la respectiva capa de material, de forma parecida al SLA, un láser dirigido con un espejo con dos grados de libertad irá sinterizando la superficie deseada de la capa del objeto, creando así capa a capa el objeto completo.

Una vez finalizada la impresión es necesario remover el excedente de polvo con un cepillo o un aspirador y realizar algunos tratamientos térmicos para liberar ciertas tensiones que se hayan podido crear en el sólido. Es frecuente encontrar la zona de sinterizado aislada, permitiendo también precalentar el material.

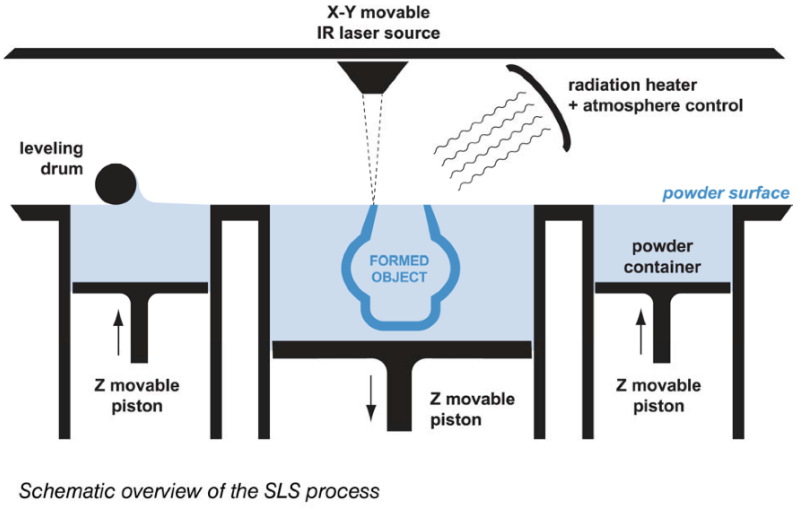


Figura 1.2 Esquema de funcionamiento de una máquina SLS o DMLS.

La diferencia principal entre ambas es que en el SLS el material que se sinteriza es plástico, nailon o ciertos tipos de cerámicas; mientras que el DMLS está orientado al sinterizado exclusivamente de ciertos metales o aleaciones como aluminio, acero o titanio.

Ambas tecnologías tienen como beneficio un gran acabado y buenas cualidades mecánicas, destacando la tecnología DMLS por el material en el que es capaz de producir objetos. Sin embargo, en cuanto a precios se refiere, las máquinas SLS parte de unos 2.000€ hasta los 15.000€, siendo aun mayor el de las impresoras DMLS raramente baja de unos 150.000€, llegando hasta unos 1.200.000€.

1.1.2 Basadas en haz de electrones

Fundición por Haz de Electrones (EBM)

EBM (Electron-Beam Melting) es una técnica similar a DMLS a diferencia de que el elemento utilizado para calentar el polvo de metal es un haz de electrones. Con esto se consigue mayor temperatura de manera que, a diferencia de DMLS, el material esta vez sí es fundido. Las piezas impresas requieren de igual manera un posterior tratamiento térmico. Económicamente se asemeja también a la tecnología DMLS, rondando las centenas de miles de euros para una de estas impresoras.

Fabricación de Forma Libre por Haz de Electrones (EBF^3)

Este planteamiento de impresión 3D fue creada en el Centro de Investigación de Langley con la idea de llevar la fabricación aditiva a entornos de gravedad cero. Sus siglas provienen del inglés Electron-Beam FreeForm Fabrication y se basa en un haz de electrones en un ambiente de vacío (menos de 10^{-4} torr) que fundirá uno o más cables, normalmente metálicos, sobre un substrato. En este caso no se hace uso de polímeros líquidos ni de polvo dado que en un ambiente de vacío es mucho mas complejo tratar con estos formatos. Con esta tecnología encontramos la posibilidad de fabricar entre otros, metales, siendo el más típico el aluminio por sus propiedades, aunque también son frecuentes aleaciones. Con esta técnica se ha llegado a conseguir fundir un material dentro de otro, haciendo posible imprimir cierto tipos de cables en el espacio.

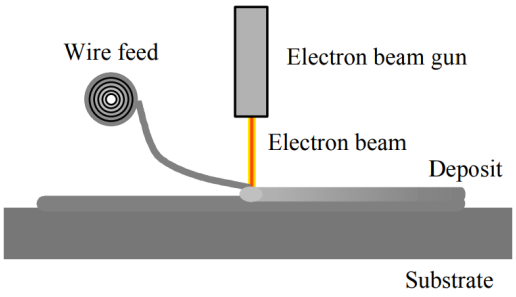


Figura 1.3 Esquema de funcionamiento de una máquina EBF^3.

No tiene sentido hacer una comparación económica con el resto de tecnologías dado que esta se restringe al alcance de ciertas empresas con capacidad de operación en ámbitos de gravedad cero y vacío.

1.1.3 Basadas en láminas

Fabricación de objetos mediante laminado (LOM)

Esta técnica de fabricación (Laminated Object Manufacturing) crea objetos tridimensionales superponiendo láminas del material elegido. Entre estos materiales encontramos papel, plástico o laminas de metal, previamente cubiertos de adhesivo. Como podemos observar en la figura 1.4, se parte de un rollo laminado (1) que se irá extendiendo sobre la superficie de impresión. Cada capa será presionada y calentada gracias a un cilindro caliente (2), consiguiendo así que junto con el pegamento que tiene la lamina se pegue cada capa a la anterior. Tras esto se recorta la capa conforme el diseño requiera (3), a veces con un láser (5) y un espejo con dos grados de libertad (4), y otras con un cúter guiado superficialmente. El conjunto de láminas pegadas y cortadas (6) baja cada capa gracias a una plataforma guiada con motores (7) mientras que la lámina de material ya cortada se va recogiendo en un nuevo rollo (8) a la vez que el todo este proceso comienza de nuevo.

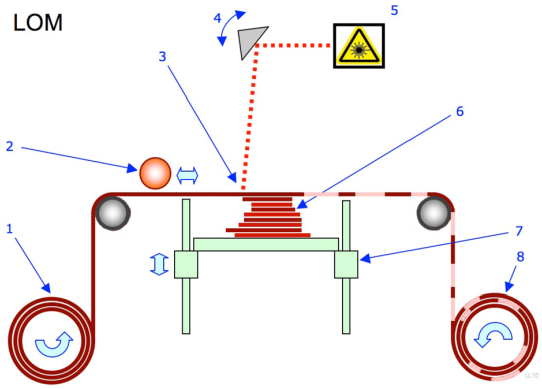


Figura 1.4 Esquema de funcionamiento de una máquina LOM.

Nuevamente encontramos como principal desventaja de esta esta tecnología el coste, dado que una máquina de este tipo puede costar de 14.000€ a 30.000€ en el mercado.

1.1.4 Basadas en extrusión de filamento fundido

Esta metodología que tiene sus comienzos en 1989 es actualmente la más extendida por todo el mundo. Su principio de funcionamiento es siempre el mismo, pero debido a su ubicuidad podemos encontrar multitud de modelos de máquinas distintas.

Fabricación Mediante Fusión de Filamento (FFF) o Modelado por Deposición de Fundido (FDM)

Comúnmente se habla de FDM (Fused Deposition Modeling) cuando hablamos de estas técnicas debido a que éste fue el nombre que le atribuye S. Scott Crump en sus orígenes, pero hoy día para evitar problemas legales con la empresa Stratasys Inc, quienes patentaron por primera vez esa técnica, se suele hacer referencia a dicha tecnología con el término FFF (Fused Filament Fabrication), acuñado por los miembros del proyecto RepRap.

El principio de funcionamiento de esta tecnología se basa en un extrusor. En éste podemos encontrar dos partes:

- En la parte superior encontramos el terminal frío. En él podemos distinguir la entrada del filamento, el cual tendrá un diámetro bien definido, el motor paso a paso que se encarga de empujar el filamento en conjunto con un rodamiento. A menudo encontramos un engranaje más grande que el del motor conectado a éste para conseguir un mayor par.
- En la parte inferior tenemos el terminal caliente o fusor. Éste contiene una resistencia encargada de calentar la boquilla. El diámetro de la boquilla define la resolución de la impresión, ya que será por ésta por donde saldrá el material en un estado semilíquido. Para realimentar el de control de la temperatura encontramos también un termopar o un termistor. Para evitar sobrecalentar el motor del terminal frío se suele colocar algún tipo de aislante o disipador entre estos dos terminales.

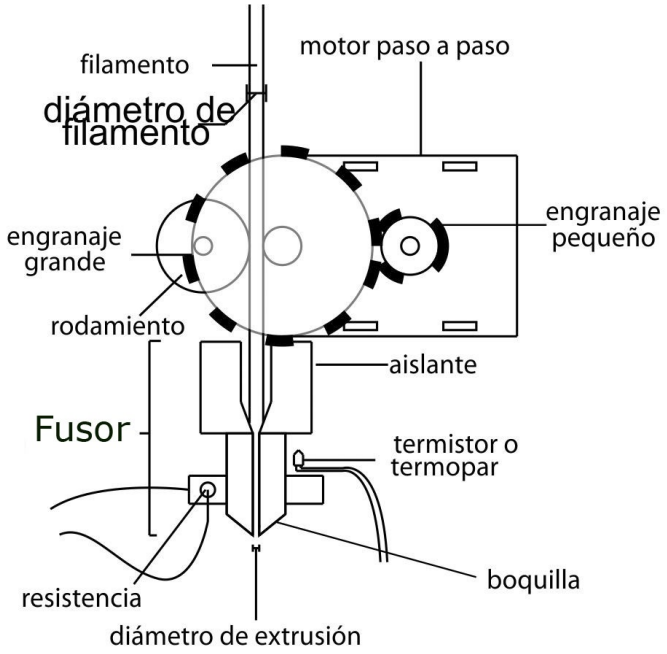


Figura 1.5 Esquema del extrusor de una impresora de tipo FFF.

Todo esto solo hace referencia a la extrusión del filamento, pero para conseguir hacer piezas es necesario mover el extrusor a lo largo del espacio de alguna manera. Antes de pasar a ver las diferentes estructuras que encontramos hoy día para ello, es importante remarcar que este trabajo se centrará en las máquinas que nacen a partir del proyecto RepRap. Se hará énfasis en éstas puesto que es a RepRap a quien se debe el auge de esta tecnología, y son estos modelos los que se encuentran con más frecuencia y en cualquier parte.

Todos los modelos tienen en común la composición de extrusor, montura y la cama. La montura es toda la estructura donde van sujetos los motores que darán movilidad al extrusor y o la cama. La cama, también conocida como cama caliente, es una plataforma lisa de dimensiones variables sobre la que se realizará la impresión. El adjetivo caliente se le suele añadir dado que esta suele ir acompañado de una plataforma de las mismas dimensiones con unas resistencias repartidas por toda la superficie que al calentarse favorecerá que el filamento se adhiera a la cama. En la gran mayoría de modelos encontraremos los siguientes 3 tipos de movimientos:

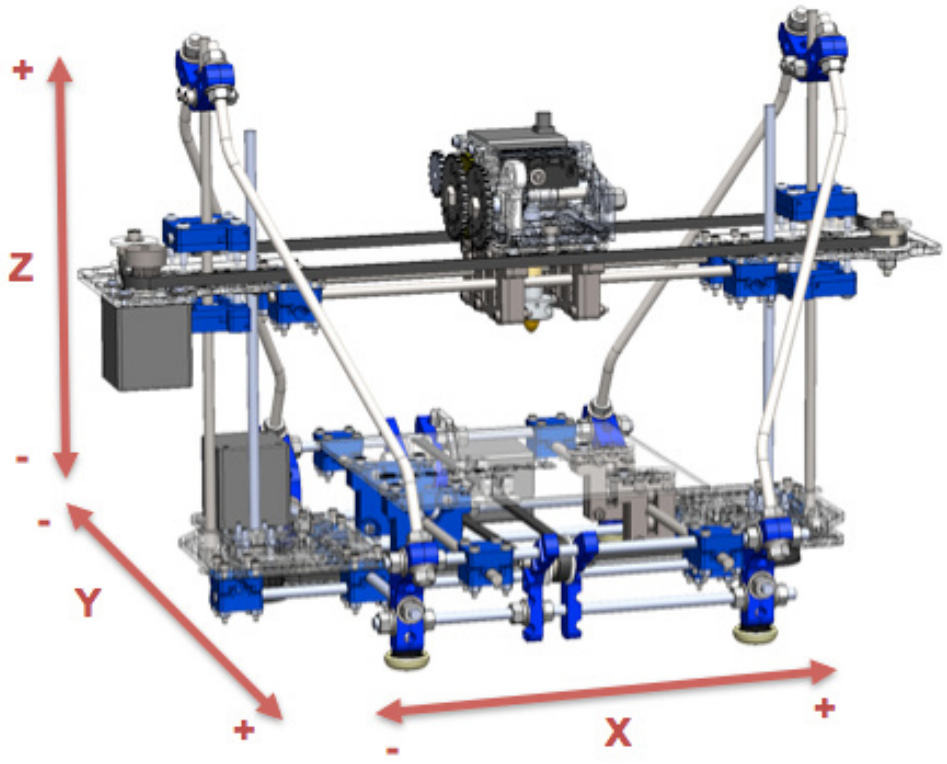


Figura 1.6 Nomenclatura de los ejes.

- El extrusor se mueve a lo largo del eje X e Y y la cama en el eje Z.
- El extrusor se mueve a lo largo del eje X y Z y la cama en el eje Y.
- El extrusor se mueve a lo largo del eje X, Y y Z y la cama permanece estacionaria.

En función de estos tipos de movimientos podemos observar los distinto modelos de impresoras que surgen del proyecto RepRap, que serán el origen de la mayoría de las impresoras que podemos encontrar hoy día. Estos tres tipos estarán distinguidos por los colores morado, verde y amarillo respectivamente (ver figura 1.7).

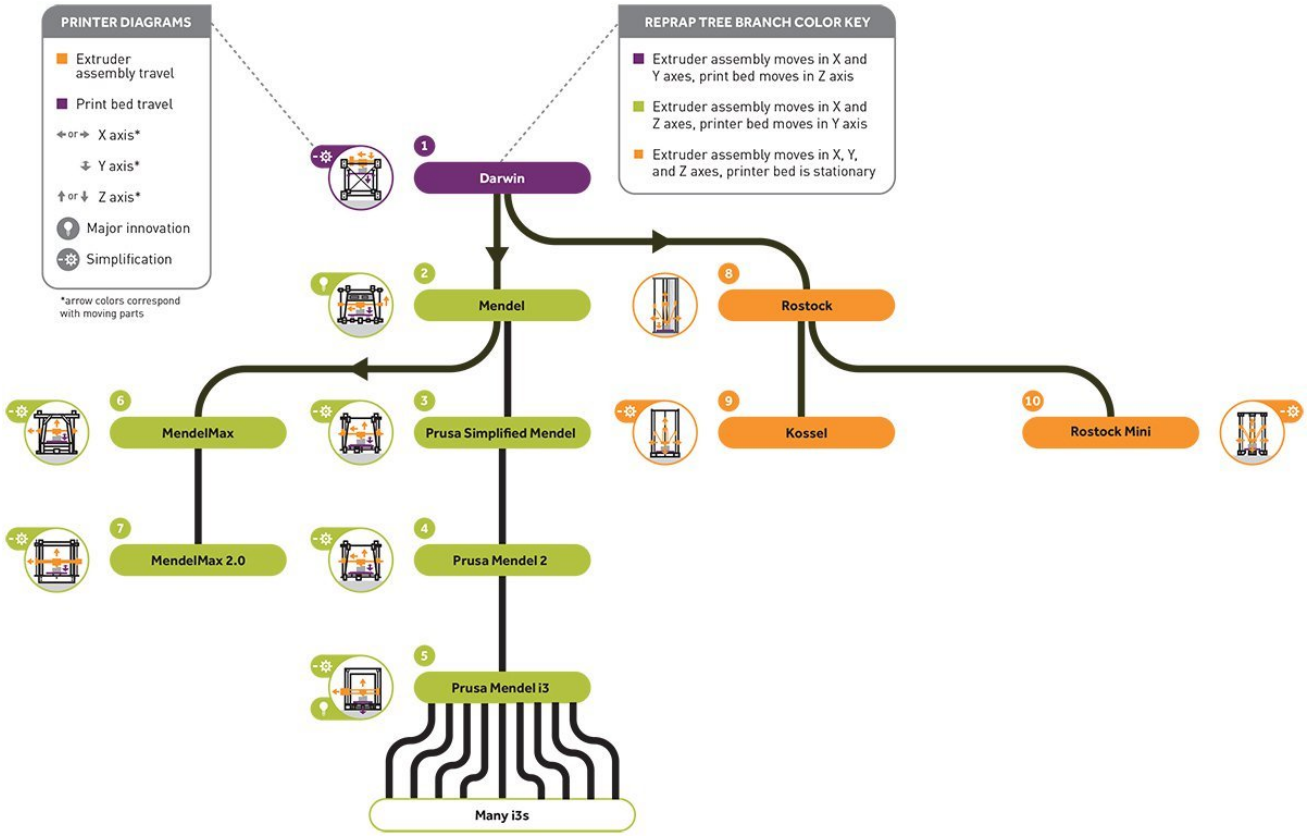


Figura 1.7 Árbol genealógico de las primeras impresoras RepRap.

Además de la variedad de modelos, también podemos encontrar variedad de filamentos para utilizar. Cada uno presentará diferentes características tales como temperatura de fusión, resistencia o flexibilidad, lo que supondrá diferentes dificultades a la hora de su uso, algunos de los cuales se afrontarán más adelante. Los filamentos más extendidos en cuanto a su uso son:

Tabla 1.1 Filamentos típicos para impresoras FFF.

Nombre	Temperatura de impresión	Cama caliente	Flexibilidad	Difucultad de uso
PLA (Ácido Poli-Lactico)	180 - 230 ºC	No necesaria	Baja	Baja
ABS (Acrilonitrilo Butadieno Estireno)	210 - 250 ºC	80 - 110 ºC	Media	Media
PET (Tereftalato de Polietileno)	220 - 250 ºC	50 -75 ºC	Media	Baja
Nailon	240 -260 ºC	70 -100 ºC	Muy alta	Media
TPE (Elastómero Termoplástico)	210 -230 ºC	No necesaria	Muy alta	Media
PC (Policarbonato)	270 - 310 ºC	90 - 110 ºC	Media	Media

Además de todos estos existen otros muchos filamentos con propiedades interesantes como lo son la solubilidad en agua, la fosforescencia, conductividad eléctrica, o imitaciones de otros materiales como maderas o metales (común mente PLA + partículas de madera o metal). Otra ventaja, a parte de la variedad de materiales que se pueden encontrar, es la facilidad para encontrar distribuidores de algunos de estos materiales en tiendas online, aun coste además asequible. Pero sin duda la mayor de las ventajas que nos ofrece esta tecnología es el precio. Es posible encontrar una amplia gama de precios que recorren desde los 120€ hasta las decenas de miles de euros.

Por otra parte, en cuanto al mantenimiento de las impresoras, se pueden encontrar multitud de fuentes de información, así como los modelos de las piezas que componen ciertas partes de la estructura de estas máquinas, haciendo posible que muchas personas puedan permitirse esta tecnología en casa.

Cualidades como el volumen de impresión, el acabado o la velocidad de impresión dependen del modelo de la impresora y de como de bien ensamblada esté la misma, ya que a menudo, cuando se adquiere una de éstas, llega completamente desmontada.

1.2 Repercusión de la impresión 3D

Las técnicas de fabricación aditivas traen de la mano un potencial cambio en el modelo de producción hasta entonces conocido. Desde la capacidad de adaptación de una industria, pasando por el modelo de desarrollo de un producto, hasta la red de transporte podría cambiar radicalmente con esta nueva tecnología. Así mismo, los campos de aplicación de ésta son innumerables (medicina, aeronáutica, educación, automoción, robótica, ingeniería civil...).

1.2.1 En la actualidad

Es difícil saber cual es la repercusión directa que la fabricación aditiva ha tenido en la sociedad. Ésta amenaza con cambiar gran parte de las cadenas de producción que sustentan los bienes que consumimos, pero aún no se ha completado ningún cambio drástico sobre ella. No obstante si podemos apreciar la aparición de nuevos espacios cada vez más extendidos por el mundo como son los FabLabs. Éstos son espacios de desarrollo colaborativos donde cualquier persona puede ir a desarrollar “casi cualquier cosa” en conjunto con el resto de gente interesada y con la ayuda de distintas maquinarias, entre las que se encuentran impresoras 3D.

Por otra parte, existen actualmente numerosos proyectos que involucran la fabricación aditiva que resultan llamativos por su alcance o sus objetivos.

Proyectos de ayuda humanitaria

Organizaciones como LimbForge o personas como Guillermo Martínez a través de la ONG Bamba Proyect se dedican a desarrollar prótesis humanas con el fin de acercarlas de manera asequible a personas desatendidas del tercer mundo. De esta forma con llevar impresoras 3D y algunos recursos más, se consigue acercar a zonas en exclusión social o poco desarrollados bienes que facilitan la vida a gente con cierta discapacidad.

Figura 1.8 Guillermo Martínez junto a un keniata y algunos de sus brazos protésicos.

Construcción de cohetes

En el ámbito aeroespacial ya hay empresas como Blue Origin o Virgin Galactic, las cuales apuestan por la fabricación aditiva a la hora de realizar diseños de cohetes. Tal es el crecimiento de este movimiento que la startup Relativity Space, fundada por entre otros Tim Ellis, antiguo miembro de Blue Origin, está trabajando en un nuevo proceso de construcción de cohetes con más de un 95 % de componentes impresos en metal. Para ello han desarrollado su propia tecnología de impresión 3D de metal con mayores dimensiones que ninguna en el mundo. En la figura 1.9 se puede observar los brazos robóticos que componen esta nueva tecnología llamada Stargate junto a un motor de propulsión creado con ella.

Figura 1.9 Motor de propulsión y sistema de impresión Stargate.

Bioimpresión

Numerosas investigaciones en la medicina actual están optando por técnicas de impresión 3D para simular cierto tipos de tejidos que imitan tejidos humanos. La mayoría de estas investigaciones siguen en curso, sin embargo ya existe un prototipo de bioimpresora 3D capaz de crear piel humana funcional. Ésta fue desarrollada por el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), la Universidad Carlos III de Madrid (UC3M) y el Hospital General Universitario Gregorio Marañón. Esta piel "puede ser trasplantada a pacientes o puede ser utilizada desde el punto de vista empresarial en el testeo de productos químicos, cosméticos o farmacéuticas ya que es producida en cantidades, tiempos y precio perfectamente compatibles para estos usos" asegura José Luis Jorcano, uno de los autores y profesor del departamento de Bioingeniería e Ingeniería Aeroespacial de la UC3M.

Construcción de puentes

Varias localidades del mundo albergan ya en sus calles puentes construidos con técnicas de fabricación aditiva. En Alcobendas, Madrid, así como en Gemert, Alemania, es posible pasear a pié, o en bicicleta respectivamente sobre puentes impresos en 3D. El primero con 12 metros de longitud y 1,75 metros de ancho y el segundo con 8 metros de longitud y 3,5 metros de ancho, ambos de hormigón.

(a) Alcobendas.

(b) Gemert.

Figura 1.10 Puentes impresos en 3D.

En Holanda se continúa trabajando todavía en la construcción mediante estas técnicas de puentes de acero, con la finalidad de poder alzar estas estructuras sobre los canales de la capital.

1.2.2 En el futuro

Cuando se habla de estas tecnologías es necesario hacer referencia a lo que puede ser una de las mayores consecuencias económica y social de este nuevo modelo de producción y es la fabricación distribuida. Ésta supone un cambio tanto en la producción como en la distribución tal y como impera hoy día. Este nuevo modelo implica pasar de grandes fabricas que distribuyen un producto al resto del mundo, a una descentralización de las fábricas, pasando de un gran centro de producción a multitud de pequeñas fábricas más cercanas al consumidor y a su vez con una producción bajo demanda de los mismos. La gran ventaja medioambiental que esto supone es el cambio de transporte físico, costoso a nivel económico y nivel medioambiental, por transporte de datos digitales. De forma que cualquier persona puede diseñar un producto en un punto A del mundo, subirlo a la nube y cualquier persona en otro punto B con acceso a dicha información podrá bajarse los archivos y fabricarlo en B.

Este modelo también acarrea desventajas como es la desaparición de multitud de puestos de empleo, con lo que no tiene sentido decir que este cambio sea un cambio totalmente positivo. Pero aun así, es imposible predecir como influirá la impresión 3D en un futuro a la economía y a la sociedad, así como es imposible de predecir cualquier cambio histórico.

1.3 Justificación del trabajo

Tras adquirir una impresora 3D de tecnología FFF, en particular el modelo Prusa I3, uno de los modelos más económicos del mercado, pude observar multitud de complicaciones asociadas a la impresión 3D.

Como se pudo observar en la tabla 1.1, existen multitud de filamentos con los que trabajar. Con tal variedad es complicado seleccionar el mejor filamento para una impresión, si bien es verdad que los filamentos más extendidos son el PLA y el ABS tanto por su disponibilidad como por su bajo precio, con el inconveniente de estar sujetos a varios problemas a la hora de su impresión, que a continuación serán tratados.

1.3.1 Problemas comunes de la impresión FFF o DFM

Las altas temperaturas a las que hay que someter la impresora así como la precisión necesaria en los movimientos hacen que las tecnologías de impresión 3D sean muy delicadas. Desde la elección del filamento hasta el lugar en el que se sitúe la impresora son factores que pueden variar mucho el resultado de la impresión. A continuación se mencionaran algunos de los problemas típicos de éstas máquinas con los que a menudo me he visto enfrentado.

Pandeo (Warping)

Uno de los problemas más frecuentes encontrados a la hora de imprimir en materiales como el ABS, el nailon o incluso el PLA en menor medida, es el pandeo, también conocido como *warping*. Éste es causado por la variación brusca de temperatura del filamento. Este cambio de temperatura produce la contracción del material, deformándolo hasta el punto de despegarse de la cama de impresión (ver figura 1.11).

Este problema se puede abordar de varias maneras:

- Subiendo la temperatura de la cama caliente, si ésta no ha llegado a su límite.
- Colocando adhesivos adicionales a la superficie de impresión.
- Aislando la zona de impresión para disminuir los cambios de temperatura.
- Imprimir una superficie auxiliar en las primeras capas.

Frecuentemente si el causante de este pandeo es el entorno en el cual esta colocado la impresora, la mejor solución es aislar la zona de impresión evitando que posibles corrientes de aire se generen y causen los indeseables cambios de temperatura.

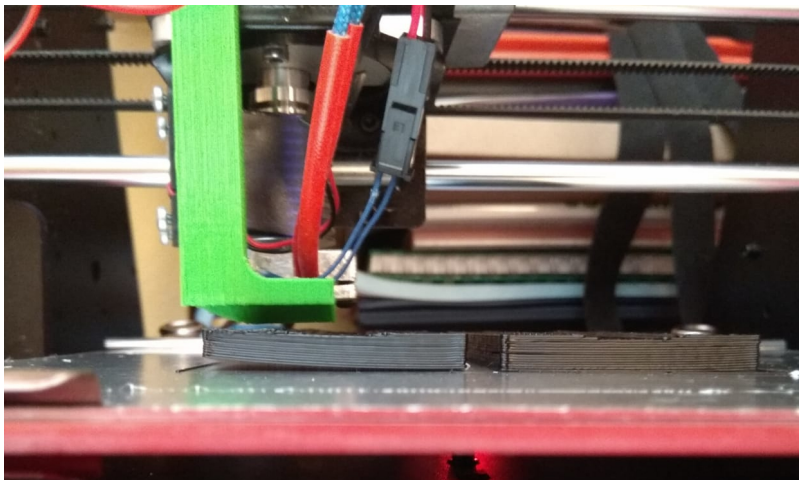


Figura 1.11 Pieza afectada por pandeo durante la impresión.

Taponamiento de la boquilla

A menudo se puede encontrar el caso de que el extrusor deja de extrudir a mitad de la impresión, causando el fallo total de la impresión. Este fallo puede provenir de distintas fuentes, una de ellas y de las más comunes es el taponamiento de la boquilla. Este taponamiento puede venir dado por una mala calidad del filamento o inconsistencias en el diámetro del mismo, temperaturas de impresión incorrectas o debido a suciedad como arena o polvo que indeseablemente entre en el extrusor. Es recomendable por tanto disponer de elementos que limpien previamente el filamento antes de la extrusión y mantener aislada la impresoras del polvo y la suciedad.

Mantenimiento

Para conseguir una impresión de calidad y evitar el sobrecalentamiento de los motores es necesario mantener la impresora en un ambiente limpio, de forma que los distintos rodamientos, correas y barras permanezcan siempre lo más limpios y engrasados posible. De lo contrario es posible que a la hora de imprimir los motores tengan que efectuar más fuerzas de lo normal, pudiendo llegar a sobrecalentarse por encima de la temperatura de funcionamiento o perder pasos durante la impresión causando que la trayectoria se desvíe de lo esperando produciendo nuevamente que la impresión falle.

Contaminación

Uno de los grandes problemas de las impresoras es la emanación de ciertos gases nocivos durante la fundición del filamento. Partículas nanométricas potencialmente nocivas para la salud se emiten mientras se extrude filamentos como el PLA o el ABS, llegando a producir un olor fuerte y bastante característico en el caso del ABS. Por ello es recomendable estar en zonas con una buena ventilación y evitar imprimir en entornos pequeños y cerrados. Este problema a menudo se solapa con el de pandeo, quedando cierto compromiso entre calidad de impresión y salud.

1.3.2 Solución propuesta

Una solución rápida y general para estos problemas mencionados es hacer un cerramiento para la impresora, que a modo de carcasa cubra a la misma. Ésta es una solución que podemos encontrar en multitud de impresoras comerciales con el inconveniente de que suelen triplicar el precio mínimo de estas máquinas. Además, el cerramiento de la impresora es una solución a la vez que un potencial problema, dado que el sobrecalentamiento de ciertos componentes de la impresora tales como la electrónica o los motores puede llegar a inhabilitarlos.

En este trabajo se propone hacer un cerramiento para la impresora 3D Prusa I3, en especial el modelo Pro W de la marca china Geeetech, así como realizar una serie de pruebas que detecten los posibles focos de futuros problemas ligado con el sobrecalentamiento de ciertos componenetes, y el posterior desarrollo de un

control de temperatura necesario para evitar futuros fallos. Para realizar este trabajo se dispone a hacer uso de las tecnologías aprendidas durante el grado tales como diseño de piezas con programas CAD, así como el uso de un microcontrolador para el monitorizado de una serie de transductores y el uso de distintos sensores y actuadores tales como sensores de temperatura, motores (ventiladores) y celdas peltier.

2 Software

A lo largo de este capitulo se hará una presentación y una justificación del software que se utilizará para el desarrollo del trabajo completo. Debido al contexto en el que se encuentra este proyecto, fuertemente influenciado por el movimiento RepRap y la comunidad del software libre, los programas han sido elegidos, en la medida de lo posible, de código abierto.

2.1 FreeCAD

FreeCAD es una herramienta de diseño paramétrico 3D creado principalmente para modelar cualquier objeto de la vida real. Este programa lleva disponible desde octubre de 2002, habiéndose escogido para este proyecto la última versión estable disponible, es decir la versión 0.17. Este programa tiene la gran ventaja de ser libre, es decir totalmente gratuito y de código abierto, lo cual permite que el desarrollo del programa se haga de forma comunitaria.

Figura 2.1 Logo de FreeCAD.

2.1.1 Licencia LGPL

Este programa esta sujeto a la licencia *GNU Lesser Gerneral Public License* en particular a la versión LGPL-2.1. Esta licencia de software libre esta orientada a librerías y fue creada como un termino medio entre *copyleft* y otras licencias más permisivas como la licencia MIT o la BSD.

Esta licencia a grandes rasgos propone la distribución y modificación libre del programa, de forma que cualquier programa que este estáticamente vinculado a esta librería deberá hacerlo bajo la licencia LGPL. En el caso de que un programa este dinámicamente vinculado, no tendrá esta restricción, pero si deberá permitir al usuario hacer ingeniería inversa para depurar y revincular la librería.

2.1.2 Compatibilidad

Una de las cualidades de FreeCAD es la capacidad de importar y exportar archivos en formatos estándar como STEP, IGES, OBJ, **STL**, DXF, SCAD, SVG, DAE, EFC, OFFm BATRAB o VRML entre otros muchos.

Esto permite una gran compatibilidad con multitud de programas, en especial en este trabajo con programas dedicados a la impresión 3D, los cuales requieren ficheros de tipo STL. Además de todos estos formatos, FreeCAD dispone de un formato de archivos propio, FCStd, que permite guardar diseños complejos ocupando muy poco espacio en memoria.

2.1.3 La comunidad

Primordialmente la elección personal de utilizar FreeCAD, a parte de fácil accesibilidad al software de forma gratuita, es debido a la comunidad de enseñanza y soporte que ésta conlleva. En Internet es posible encontrar multitud de tutoriales en los cuales se puede aprender a manejar estos programas de manera fácil y accesible.

La comunidad hispanohablante goza además del apoyo de Juan González Gómez, también conocido como Obijuan, ingeniero de Telecomunicaciones y doctor en Robótica, quién actualmente trabaja como profesor/investigador en la Universidad Rey Juan Carlos de Madrid. Obijuan es uno de los creadores del grupo Clone Wars dentro de la comuidad RepRap, quienes documentan todo lo necesario para construir una impresora 3D en castellano,así como también es el creador de lo que él llama la Obijuan Academy, en la cual se puede encontrar multitud de información para aprender las herramientas básicas de FreeCAD para diseñar cualquier objeto.

2.1.4 Instalación

Otro beneficio de de este programa es la facilidad para instalar este programa, estando disponible para los sistemas operativos más utilizados: Linux, Windows y MacOS.

Para instalar FreeCAD en Linux de manera sencilla hay dos opciones:

- Por línea de comando:

```
$ sudo apt-get install freecad
```

- Descargando la imagen de la wiki oficial.

```
https://www.freecadweb.org/wiki/Download
```

Ésta ultima opción no necesita ni siquiera una instalación como tal, es un archivo *.AppImage* que contiene el programa completo.

Para instalarlo tanto en Windows como en MacOS se hará de forma trivial, descargando el correspondiente ejecutable del link de la wiki oficial que se encuentra justo arriba.

2.2 Repetier-Host

Repetier-Host es uno de los programas más extendidos entre las máquinas RepRap para el control de éstas. Éste software es de los más extendidos debido a su gran compatibilidad con la mayoría de impresoras derivadas del proyecto RepRap y por su interfaz intuitiva y fácil de usar. La conexión se puede hacer por comunicación serie mediante un cable USB o a través de internet (TCP/IP).Este programa dota al usuario de la capacidad de colocar diferentes objetos, copiarlos, moverlos, escalarlos o rotarlos dentro de la cama virtual, análoga a la de la impresora, para su posterior impresión.

Figura 2.2 Logo de Repetier-Host.

Este programa disponen de distintos *slicers*, que son "subprogramas" encargados de pasar los objetos a trayectorias válidas para la impresora, las cuales vendrán en formato G-Code (lenguaje de descripción de trayectorias para CNCs). Tanto en la configuración de Repetier-Host, como en la configuración del *slicer* que se esté usando, se encuentran multitud de parámetros que permiten: calibrar debidamente la impresora, configurar las temperaturas necesarias para cada tipo de filamento, cambiar la velocidad de impresión, cambiar la densidad de relleno de los objetos, etc.

Repetier-Host es gratuito y está disponible para Linux, Windows y MacOS. Su instalación es trivial en todos los sistemas operativos y están disponibles en el siguiente enlace:

```
https://www.repetier.com/download-now/
```

2.3 Code Composer Studio

Code Composer Studio o CCS es un entorno de desarrollo integrado (IDE) específico para desarrollar aplicaciones en procesadores *embedidos* de Texas Instrument. Entre estos procesadores podemos encontrar DSPs, OMAPs SoCs, microcontroladores Hercules, **MSP430** o microcontroladores de la serie Tiva por ejemplo.

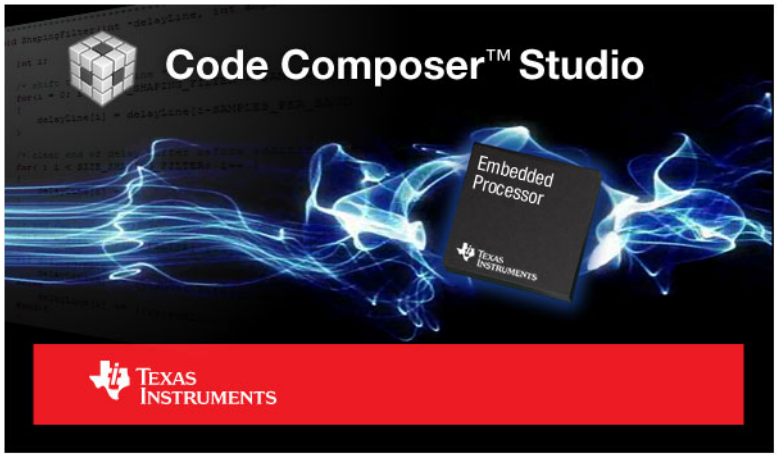


Figura 2.3 Imagen de inicio de CCS.

Para este trabajo se hará uso de la versión 6.2 de este software. Esta herramienta nos proporciona el entorno completo para desarrollar y depurar aplicaciones de bajo nivel, también conocidas como *baremetal*

(metal-desnudo), en las cuales las instrucciones se plasman en lógica hardware sin un sistema operativo entre procesador y aplicación. Se ha seleccionado este IDE para programar un microcontrolador MSP430G2553, cuyas siglas significan Procesador de Señales Mixtas (Mixed-Signal Procesor).

2.3.1 Grace

Grace es otra herramienta de TI, en este caso enfocada al desarrollo de aplicaciones con microcontroladores de la familia MSP430, gracias a la cual se puede realizar toda la configuración inicial de los distintos periféricos disponibles de forma visual y sencilla. Esta herramienta está disponible como plug-in del propio CCS hasta la versión 6.1.2, estando también disponible la versión independiente, dando cobertura a las versiones superiores.

2.3.2 Instalación

Nuevamente este programa esta disponible para Linux, Windows y MacOS. Podemos encontrar los archivos de instalación en el siguiente enlace:

```
http://processors.wiki.ti.com/index.php/Download_CCS
```

La instalación tanto en Windows como en MacOS es de nuevo trivial. En el índice se encuentran las versiones disponibles, siendo la escogida para este caso la versión 6.2.0.00050.

Para la instalación en Linux serán necesarios una serie pasos adicionales:

- Descargar el archivo comprimido en .tar.gz.
- Descomprimir el archivo:

```
$ tar -xvzf <nombre del archivo>.tar.gz
```

- Dentro del directorio donde se hayan descomprimido los archivos, ejecutar el binario de instalación:

```
$ ./ccs_setup_linux64_X.X.X.XXXXX.bin
```

- Seleccionar el directorio donde se quiera hacer la instalación y presionar *Finish*.
- Una vez instalado, nos dirigimos a: $< directorio\_de\_instalacion > /ccsvX/install\_scripts$ e instalar los drivers necesarios como root.

```
$ sudo ./install_drivers.sh
```

Nótese que hay que reemplazar las ‘X’con el número correspondiente a la versión que se esté instalando.

2.4 PuTTy

PuTTy es otro programa de código abierto, el cual ofrece un cliente SSH, rlogin, TCP así como una consola de comunicación serial. Gracias a este programa podremos comunicarnos con el MSP430 mediante puerto serie y logear toda la información recibida en un fichero para su posterior tratamiento.

Figura 2.4 Logo de PuTTy.

Este programa con una interfaz gráfica muy simple dotará al usuario de multitud de configuraciones para dicha terminal. Para su instalación en Windows tan solo habrá que dirigirse a:

```
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
```

Descargar la versión correspondiente e instalarlo. En ubuntu su instalación es aun más sencilla teniendo únicamente que escribir en la terminal:

```
$ sudo apt-get install putty
```

3 Hardware

Para el desempeño de este trabajo se ha recurrido a diferentes elementos físicos que recorren desde un listón de madera hasta un microcontrolador MSP430G2553. En este capítulo se afrontará la descripción de los distintos elementos la justificación de su uso y el presupuesto que suponen, ya que como se ha observado, el resto de elementos no físicos son en su totalidad gratuitos, y por tanto el coste total del proyecto se confronta en este apartado.

3.1 Estructura

Uno de los principales requisitos para la elección de los materiales que confrontarán el cerramiento será la temperatura. En segundo lugar, este trabajo persigue ser asequible económicamente, buscando que esta solución sea útil para todas las personas que persigan solventar los problemas mencionados en el apartado 1.3.1, así como cualquier otro que requiera una solución semejante.

3.1.1 Poliestireno de propósito general (GPPS)

Para las 5 paredes que cubrirán la impresora 3D se ha escogido poliestireno o GPPS (General Purpose PolyStyrene). Este material nos ofrece numerosas ventajas, entre ellas que es transparente, lo que nos permitirá ver el transcurso de la impresora en todo momento y desde cualquier ángulo. Esto puede ayudar a detectar cualquier problema que suceda durante la impresión evitando que éste llegue a producir fallos irreversibles. Alguna de sus propiedades físicas más interesantes son:

Tabla 3.1 Propiedades interesantes del GPPS.

Propiedad	Unidad	Valor/Rango
Temperatura de fusión	ºC	180 - 260
Densidad	g*m/L	1.04 - 1.05
Conductividad térmica	W/m*K	0.35
Temperatura de trabajo en régimen permanente	ºC	-73 - 82
Precio unitario (2mm de grosor)	€/m^2	16

Entre estas propiedades cabe destacar la temperatura de trabajo en régimen permanente, la cual marcará el límite de temperatura que podrá soportar la estructura.

Figura 3.1 Símbolo internacional del poliestireno.

Se ha optado por utilizar 5 planchas de 50x50x0.2 cm para formar un cubo sin base. El precio unitario de cada plancha es de 4€ que hacen un total de 20€.

Alternativas

La elección del poliestireno viene dada por la facilidad para encontrar este material en tiendas cercanas, así como por su resistencia al calor. En su defecto existen multitud de plásticos que pueden cumplir los mismos requisitos. Planchas de policarbonato o metacrilato pueden ser usados en su lugar.

El espesor de las planchas no necesariamente ha de ser de 2mm. Es recomendable el uso de espesores iguales o menores a éste para evitar sobrecargar el marco de la impresora.

3.1.2 Listón de madera

Para dotar de mayor rigidez a la estructura y soportar el peso de las planchas de poliestireno fronteras y traseras se utilizarán 2 listones de maderas de 52 cm de longitud. El perfil de la sección ha de ser preferentemente rectangular para facilitar la unión entre estos y la plancha frontal, la cual será abatible. En mi caso personal se usarán unos listones provenientes del embellecedor de un estore de madera, los cuales tienen un perfil distinto (ver figura 3.2), para reutilizar el material disponible en casa.

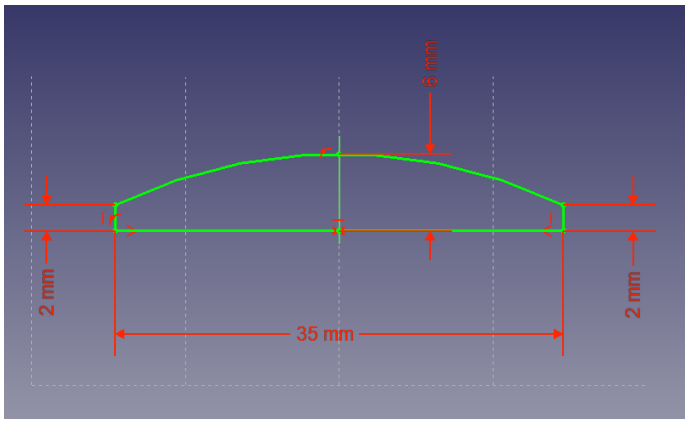


Figura 3.2 Perfil de los listones de madera.

El precio de total de los listones de madera de dimensiones parecidas se encuentra entre 1,5€ y 3€.

Alternativas y observaciones

Otros materiales tales como el metal pueden ser seleccionados para la construcción de este cerramiento. El requisito que han de cumplir es que tengan la suficiente resistencia como para soportar el peso de las 2 planchas que hayan sido seleccionadas.

Si se usa un perfil distinto, será necesario cambiar el modelo de la abrazadera en FreeCAD, debiendo para ello modificar únicamente el sketch *PerfilMadera* que se encuentra en *Pieza_Final>Pieza_sin_tornillos>BarraMadera*.

3.1.3 Tornillos

Para fijar debidamente las piezas impresas que se expondrán a continuación serán necesarios 4 tornillos de 2x60mm y otros 4 de 2x20mm con sus correspondientes 8 tuercas de 2mm. También es posible reutilizar cualquier tornillo que tenga dimensiones parecidas y fijen debidamente la estructura. El precio de estos tornillos puede rondar 1,20€.

3.1.4 Imanes

Se hará uso de unos pequeños imanes que fijen la parte delantera en posición vertical. Dos tipos de imanes serán utilizados, ambos cilíndricos de 5mm de diámetro, unos con 4mm de altura y otros dos de 10mm. Su precio es de 50 céntimos la unidad, siendo necesario dos de cada tipo.

3.1.5 Disipadores

Para mejorar el rendimiento de las celdas peltier se hará uso de disipadores. Gracias a ellos el salto de temperatura que se cree en las celdas se hará en base a la temperatura a la que se encuentre cerramiento. Sin ellos, el salto de temperatura creado en la celda haría que el lado caliente llegara a temperaturas de hasta 90ºC, desperdiciando una considerable cantidad de energía en calentar innecesariamente uno de los lados de las celdas.

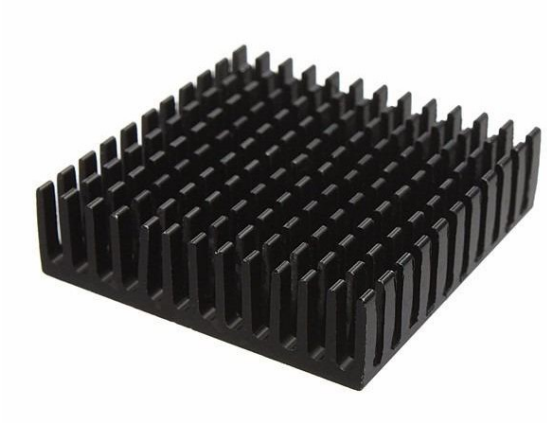


Figura 3.3 Disipador de aluminio.

Gracias a los disipadores y los ventiladores, administrando una corriente no superior a los 2 o 3 amperios se conseguirá una transferencia de calor suficiente para mantener los motores a una temperatura segura. De lo contrario sería necesario añadir una fuente de alimentación y ciertos elementos de regulación de corriente capaces de subministrar hasta 6 amperios por celda. Estos elementos se podrán sacar de antiguas torres de ordenadores o ser compradas por menos de un euro la unidad.

3.1.6 Piezas impresas

Una de las grandes ventajas de trabajar con una máquina de fabricación aditiva es la posibilidad de fabricar sin mucho esfuerzo ni coste todas las piezas que sean necesarias para ensamblar el cerramiento. A continuación se verán las especificaciones de las piezas que se harán uso en este diseño. Todos los diseños se adjuntarán tanto en formato *.FCStd* para su modificación, si fuera necesaria, como en *.stl* para su impresión.

Pinza para el marco de la impresora

Con motivo de hacer abatible la cara delantera del cerramiento se colgarán tanto la cara delantera como la trasera del marco de la impresora. Para ello será necesario apoyar dos listones de madera sobre el marco para posteriormente poder colgar de estos la cara delantera. Por ello, se ha diseñado una pieza, la cual se encaje a modo de pinza sobre los extremos del marco de la impresora, para evitar que esta carga añadida afecte a ninguna parte móvil.

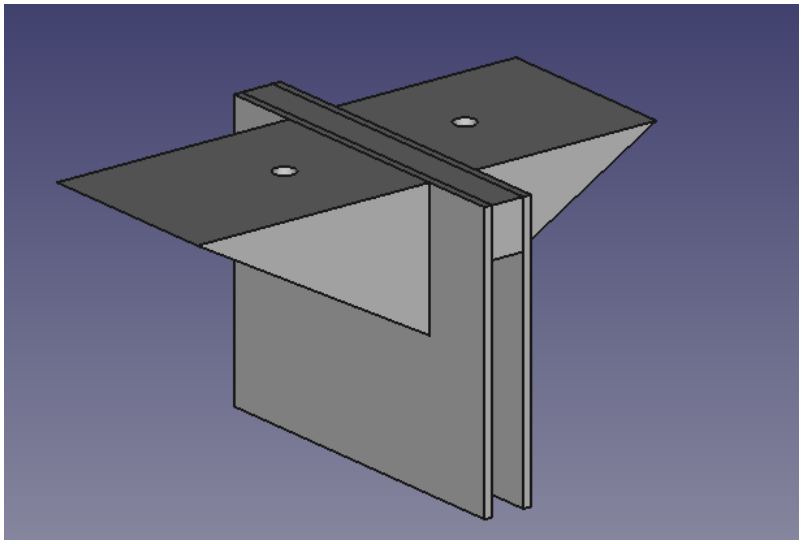
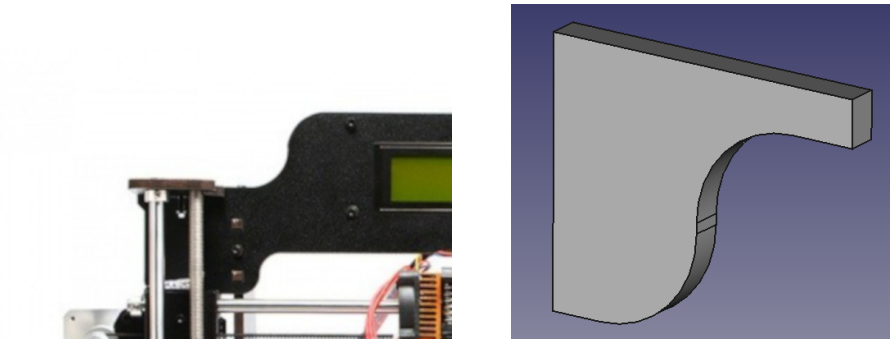


Figura 3.4 Pinza derecha del marco.

Sobre esta pinza irán la cara superior del cerramiento, un listón de madera y una abrazadera que se atornillará a la misma. Los rasgos más característicos de esta pieza es la forma curva del interior de la pinza para ajustarse a la curva que dispone el marco del modelo de impresora que se esta tratando. Por otra parte, los voladizos que salen de la pieza como se pueden observar son asimétricos dado que los listones de madera no irán apoyados sobre el centro de estos, sino que estarán ligeramente sobresaliendo hacia adelante. Además, éstos tendrán forma triangular, optimizando el material empleado y su resistencia ante el momento flector al que será sometido.



(a) Marco de la impresora. **(b)** Interior de la pinza.

Figura 3.5 Interior de la pieza.

Esta pieza será impresa en ABS aprovechando su alta resistencia al calor y su resistencia mecánica. La configuración para la impresión de esta pieza será la siguiente.

Se adjunta con este trabajo el diseño de ambas esquinas bajo los nombres de *Pinza_Esquina_Derecha* y *Pinza_Esquina_Izquierda*.

Tabla 3.2 Configuración de impresión para ABS.

Parámetro	Unidad	Valor
Temperatura de la cama	ºC	110
Temperatura de la boquilla	ºC	245
Velocidad de impresión	mm/s	60
Velocidad de perímetro exterior	mm/s	50
Velocidad de relleno	mm/s	100
Densidad de relleno	%	20

Abrazadera

Para asegurar que ni el listón de madera ni la cara superior se muevan de la pinza del marco, se ha diseñado una pieza encargada de sostener el listón de madera firme y recto, para así afianzar la estructura al marco.

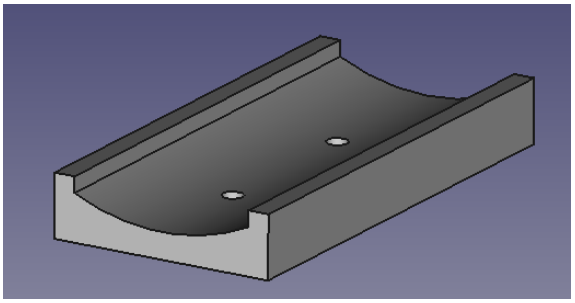


Figura 3.6 Abrazadera.

Como se mencionó anteriormente, esta pieza dependerá del listón que se seleccione. Aprovechando la jerarquía de dependencia de FreeCad se podrá cambiar fácilmente a la necesidad de cada uno. Nuevamente esta pieza será impresa en ABS, siendo posible cambiar el material dado que esta pieza quedara fuera del cerramiento y no será necesario someterla a altas temperaturas. La configuración para esta impresión es la misma utilizada para la pinza que se puede encontrar en la tabla 3.2.

Escuadras traseras

La unión entre las planchas laterales y la trasera se hará mediante 4 escuadras con un hueco de 2mm. Dos de ellas irán en la parte superior y 2 en la parte inferior. Si se ha escogido otro grosor para las planchas de PS, deberá cambiarse el ancho de las dos planchas que estarán bajo la ruta *Pieza_final>Planchas*.

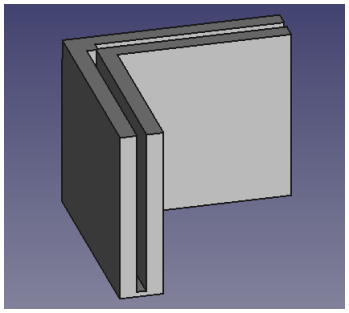
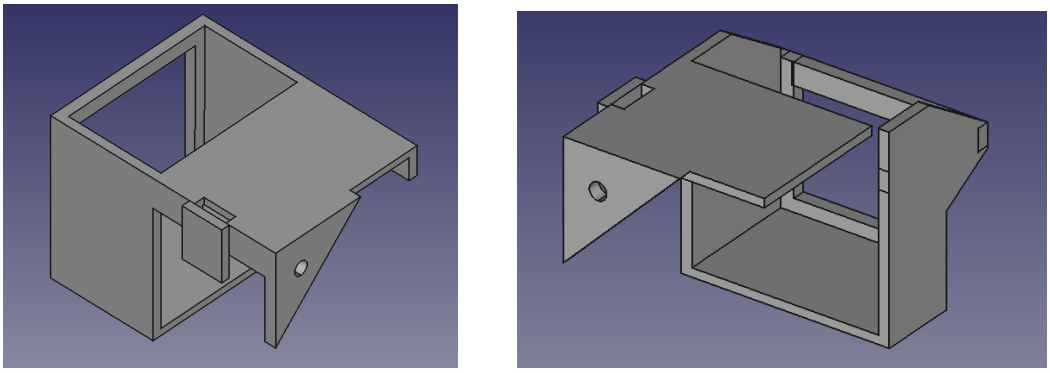


Figura 3.7 Escuadra trasera.

Se ha añadido un 10 % al grosor de estas planchas para evitar posibles fracturas por desperfectos de la impresión. Para esta pieza se usarán nuevamente los valores de la tabla 3.2 para su impresión en ABS.

Soporte para los motores

Ya que se busca refrigerar los motores más comprometidos con celdas peltier será necesario un soporte en el cual se puedan poner tanto el sensor de temperatura, como el conjunto de celda peltier, disipador y ventilador. Dado que cada uno de los motores esta sujeto a distintas restricciones por su posición, serán necesario 2 diseños ligeramente diferentes.



(a) Eje X. **(b)** Eje Y.

Figura 3.8 Soporte para motores.

La principal diferencia entre el diseño del eje X y del eje Y será el compromiso de espacio que tiene el motor del eje Y el cual se encuentra bajo la cama caliente. Será necesario eliminar algunas partes del soporte del segundo motor para evitar la obstrucción el paso de la cama caliente y conseguir que encaje teniendo en cuenta la proximidad de una de las tuercas de la estructura.

En ambos casos el motor dispondrá de un hueco para un tornillo disponible, el cual se usará para asegurar la estructura. En la ranura de la parte izquierda irá colocado el sensor de temperatura de forma que mida la temperatura del cuerpo de los motores. La parte posterior estará medida para que encajen justo la celda peltier junto con el disipador y un ventilador de las medidas especificadas. En caso de cambiar alguno de los elementos se deberán modificar los soportes. Los soportes irán nuevamente impresos en ABS, por su compromiso con la temperatura, según los parámetros indicados en la tabla 3.2.

Soporte para imanes

Para asegurar que el panel delantero se mantenga cerrado se hará uso de unos pequeños imanes. Para conseguir sujetar estos debidamente se han diseñado unas pequeñas pinzas que se colocaran en las esquinas inferiores de los laterales y del panel frontal.

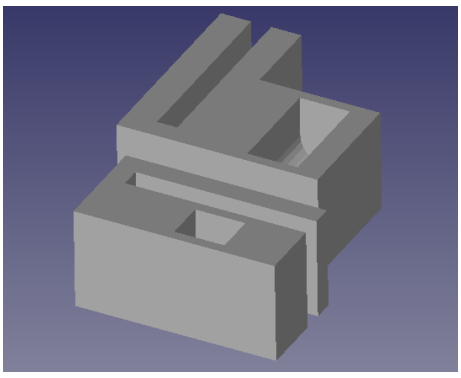


Figura 3.9 Pinzas para los imanes de la esquina frontal izquierda.

Estas piezas tienen un hueco de 2mm de grosor para ajustarse a las planchas de poliestireno, así como un hueco para los imanes, de la dimensión descrita en el apartado 3.1.5. Dado su localización lejos del foco de calor, se podrá imprimir en el material que se crea conveniente, siendo utilizado en mi caso por comodidad ABS con la configuración utilizada hasta ahora.

Bisagras

Para articular la cara frontal se utilizará un filamento flexible para imprimir un par de bisagras que sirvan de unión y de articulación entre los listones de maderas y la plancha de PS delantera.

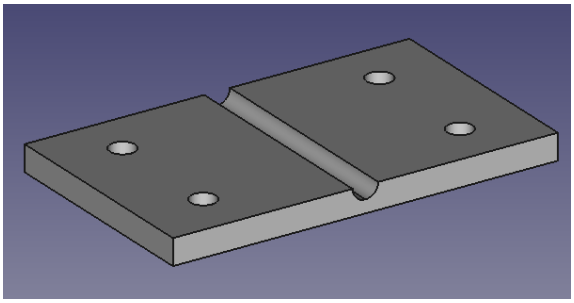


Figura 3.10 Bisagra.

La configuración necesaria para imprimir las bisagras con el filamento flexible es:

Tabla 3.3 Configuración de impresión para filamento flexible.

Parámetro	Unidad	Valor
Temperatura de la cama	ºC	60
Temperatura de la boquilla	ºC	240
Velocidad de impresión	mm/s	20
Velocidad de perímetro exterior	mm/s	15
Velocidad de relleno	mm/s	20
Densidad de relleno	%	10
Habilitar retracción	-	No

3.2 Electrónica

Durante una impresión normalmente tanto la boquilla como la cama caliente permanecerán encendidas y a unas temperaturas elevadas. Esto puede producir problemas a la hora de mantener la impresora aislada, ya que la electrónica que esta presente en la impresora puede sobrecalentarse hasta llegar al fallo permanente. Por ello surge la necesidad de introducir cierta circuitería de control de temperatura para asegurar que el entorno del cerramiento, y cada uno de los elementos vulnerables de la impresora se mantienen a temperaturas aceptables

3.2.1 Sensores de temperatura LM35DZ

Para poder realimentar el microcontrolador con valores de temperatura, será necesario tener transductores que transformen la temperatura a una tensión para realizar el cómputo necesario. En este caso se ha optado por sensores LM35DZ. En la tabla 3.4 podemos ver sus características más relevantes.

Figura 3.11 Aspecto del sensor LM35DZ.

Tabla 3.4 Características del sensor LM35DZ.

Característica	Unidad	Valor/Rango
Precisión	ºC	$\pm$0.9
Rango de funcionamiento	ºC	0 - 100
Sensibilidad	mVºC	10
Dimensiones	mm	5,21x4,19x5,34
Tipo de encapsulado	-	TO-92
Precio unitario	€	1,72

Con este sensor, el cual es capaz de trabajar bajo las condiciones que marca el poliestireno (80ºC), se podrá realizar el sensado de las diferentes zonas de interés del cerramiento. Con un total de 3 sensores, el dinero necesario para ese elemento será de 5,16€. En su defecto, cualquier sensor de temperatura que se estime oportuno y funcione en el rango de los 20ºC-80ºC será válido.

3.2.2 Celdas peltier TEC1-12706

Refrigerar los motores paso a paso en un entorno con temperaturas elevadas no es tarea fácil. Será necesario por tanto introducir elementos con una alta conductividad térmica, capaces de reducir la temperatura por

debajo de la temperatura a la que se encuentre el entorno.

Figura 3.12 Celdas peltier TEC1-12706.

Para ello se hará uso de celdas peltier, las cuales están compuestas por varias uniones semiconductoras P-N, que al pasar por ellas una corriente eléctrica, transportan el calor de una de sus caras a la otra. A este efecto se le conoce como efecto Peltier debiendo su nombre al físico francés Jean Peltier. Estas uniones semiconductoras tienen además un encapsulado cerámico, el cual les proporciona una mayor conductividad térmica.

Una gran ventaja de las celdas peltier como elemento de refrigeración es que permite regular el flujo de calor producido regulando la alimentación de las mismas, de forma que se puede conseguir un control más eficiente. En particular se ha elegido el modelo TEC1-12706, cuyas características destacables son:

Tabla 3.5 Características de la celda TEC1-12706.

Característica	Unidad	Valor/Rango
Potencia de transmisión de calor	W	60
Temperaturas de operación	ºC	-30 - 83
Diferencia de temperatura máxima	ºC	66-75
Dimensiones	mm	40x40x3,8
Corriente máxima	A	6,4
Voltaje máximo	V	16.4
Voltaje nominal	V	12
Precio unitario	€	2,70

Con 2 de estos dispositivos será suficiente para refrigerar los motores más comprometidos (eje X e Y).

3.2.3 Ventiladores

Tanto para la entradas como la salidas de aire, y la adecuada refrigeración de las celdas peltier se hará uso de ventiladores. En mi caso serán pequeños ventiladores de 40x40x10mm sin escobillas con una tensión nominal

de 12V, con un valor de 2,85€. En la medida de lo posible se ha intentado reutilizar todos los ventiladores posibles de ordenadores antiguosde torres y portatiles.

Figura 3.13 Ventilador sin escobillas de 40x40x10mm.

Se hará uso de 2 ventiladores, los cuales junto a disipadores ayudaran a fijar el lado caliente de las celdas peltier a temperatura ambiente, mejorando su rendimiento de forma notable.

3.2.4 Módulo L298N

Para el control de las celdas peltier y los ventiladores asociados a éstas se propone usar 2 puentes en H con el circuito L298N. Gracias a este módulo podremos hacer un control sencillo de la corriente que subministramos a las celdas peltier mediante señales PWM. Este módulo es interesante también dado que será capaz de aguantar hasta 3 A en corriente continua, lo cual sera suficiente teniendo en cuenta que la celda peltier esta refrigerada por aletas en el lado caliente.

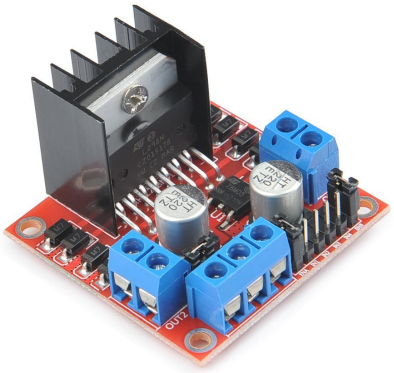


Figura 3.14 Módulo L298N.

Podemos encontrar este módulo desde los 3,30€, siendo necesarios una unidad, ya que en un solo módulo L298N encontramos 2 puentes H completos.

3.2.5 Microcontrolador MSP430G2553

Ante la necesidad de controlar diferentes elementos tales como los puentes en H y hacer la lectura analógica de diferentes sensores de temperatura, todo esto relacionado en un bucle de control, se ha escogido en microcontrolador MSP430G2553 como elemento de control. Este microcontrolador perteneciente a la subfamilia MSP430G2X, dentro de la familia MSP430, tiene características muy interesantes tales como un una velocidad de procesamiento de 16MHz, hasta 8 canales de entrada para un conversor analógico-digital, timers con los que se podrán simular señales PWM, comparadores e interfaces de comunicación I2C, SPI y UART. Otro aspecto interesante son los modos de bajo consumo disponible, los cuales, mediante una programación adecuada, pueden dotar de gran autonomía al microcontrolador con pequeñas baterías.

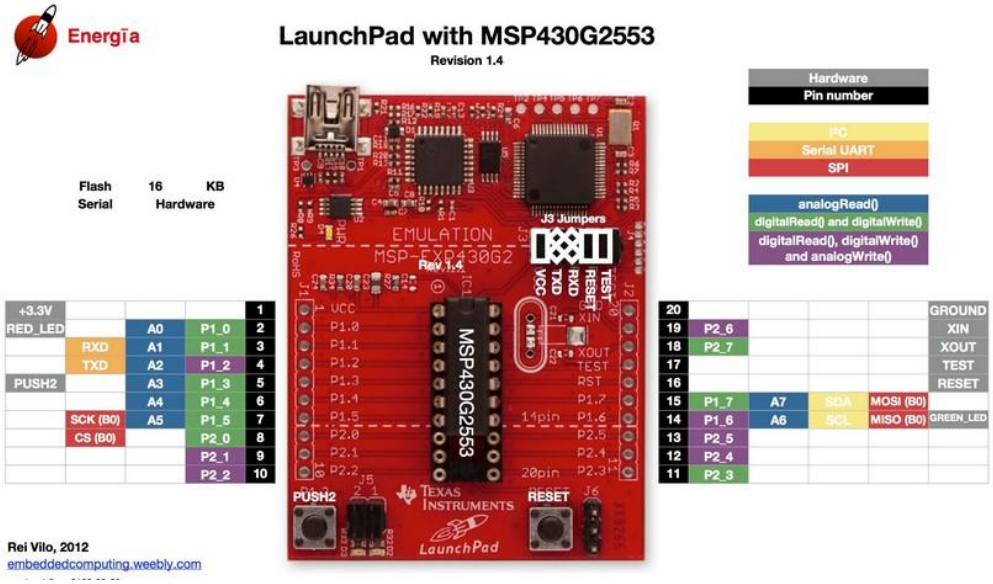


Figura 3.15 MSP430G2553 LaunchPad junto a algunas características.

Se podrá programar el microcontrolador usando varias herramientas tales como CCS, Grace o Energía (IDE similar al de Arduino), permitiendo utilizar más a fondo todas las capacidades del MSP430 con CCS o realizar aplicaciones más sencillas de manera rápida con Energía.

Se utilizará el microcontrolador con el LaunchPad el cual nos permite programar y depurar las aplicaciones de manera sencilla. La elección este micro en concreto se debe principalmente a 3 motivos. El primero, un motivo técnico, y es que posee suficientes entradas analógicas para el desempeño de este trabajo. En segundo lugar, por un motivo económico, y es que este microcontrolador con el LaunchPad se puede encontrar online desde unos 9€, precio más que razonable para las prestaciones que este nos ofrece, y por último, por una razón académica, y es dotar de cierta coherencia al este trabajo junto con la materia y las prácticas superadas durante la carrera.

3.3 Presupuesto

Haciendo un recuento en general de todos los elementos necesarios que no se pueden fabricar con la impresora, se obtiene un presupuesto total de 58,92€.

Tabla 3.6 Presupuesto.

Elemento	Unidades	Precio unitario (€)	Dimensiones (mm)	Precio total (€)
Plancha de poliestireno	5	3,95	500x500x2	19,75
Listón de madera	2	0,8	520x35x4	1,6
Tornillo T 2x20	4	0,05	2x20	0,2
Tornillo T 2x40	4	0,1	2x40	0,4
Tuerca	8	0,05	2	0,4
Imán cilíndrico	2	0,5	5x4	1
Imán cilíndrico	2	0,5	5x10	1
Disipadores	1	0,63	40x40x11	0,63
Sensor LM35DZ	4	1,72		6,88
Celda peltier TEC1-12706	2	2,7		5,4
Ventilador	2	2,85	40x40x10	5,7
Módulo L298N	1	3,3		3,3
MSP430G2553	1	9,71		9,71
Cables	40	0,02375		0,95
Protoboard Mini	2	1		2
Total				58,92

4 Desarrollo

Una vez visto todos los elementos necesarios para la realización del trabajo se pasará a describir como se ha ido realizando el trabajo completo. Se comenzará describiendo el proceso de construcción del cerramiento físico, posteriormente se mostrarán las pruebas realizadas con el cerramiento y se pasará a el montaje del sistema de control de temperatura.

4.1 Construcción del cerramiento

En este apartado se abordará toda la información relativa al cerramiento. Se tratarán las decisiones tomadas para su diseño, así como todos los pasos necesarios para construir desde cero la totalidad de la estructura.

4.1.1 Diseño

El primer paso de este trabajo, previo a la construcción del cerramiento, es el diseño del mismo. El diseño desde un primer momento está motivado por la simplicidad, ya que se busca que el sistema sea relativamente fácil de transportar, de montar y económico. Dado que esta parte del trabajo es la más sencilla, se ha intentado comprar el mínimo material posible para obtenerla.

Las principales restricciones a las que está atada la estructura son la resistencia a la temperatura y que el cerramiento sea transparente, para facilitar la supervisión de la impresora durante todas las pruebas, así como durante las impresiones. Para ello se eligieron planchas de poliestireno transparente de 2 milímetros de grosor, ya que este es un material fácil de encontrar, resistente a temperaturas elevadas y económico.

Para hacer abatible la parte frontal del cerramiento se ha optado por hacerla abatible hacia arriba, por la falta de espacio de la habitación en la que esta está alojada. Por ello y para evitar sobrecargar la estructura de PS y la falta de rigidez, se decidió que tanto la plancha superior, la delantera y la trasera irían apoyadas sobre el marco de la impresora, más específicamente, de dos listones de madera que están apoyados sobre el marco.

Una vez que la estructura cobra forma, se diseñaron las piezas necesarias para ensamblar todas las partes. Por una parte sendas pinzas para el marco junto con las abrazaderas fijarán ambos listones al marco. Para los laterales se diseñaron las escuadras que fijarán ambos lados a la parte trasera. Por último, para evitar comprar las bisagras que den movilidad a la cara delantera, se diseñaron a medida de los listones de madera.

4.1.2 Montaje

Una vez que se obtienen las 5 planchas de PS de 500x500, será necesario realizar algunos cortes y orificios. Para ello se puede recurrir a varias soluciones, como cortadoras láser, pedir las planchas con el corte a medida o realizarlo a mano con un cúter y un pequeño taladro. Estas soluciones a su vez dependerán del material elegido en caso de cambiar el PS de 2 milímetros por otro. En el apéndice A se pueden encontrar los planos con todas las medidas necesarias para realizar los cortes y orificios necesarios a las diferentes planchas de PS.

Una vez que todos los paneles están listos, será necesario imprimir las piezas propuestas. A continuación en la tabla 4.1 se describen todas las piezas necesarias y el material recomendado.

Tabla 4.1 Piezas impresas necesarias para el cerramiento.

Nombre de la pieza	Cantidad	Material	Tiempo de impresión estimado por unidad (min)
Pinza derecha para el marco	1	ABS	50
Pinza izquierda para el marco	1	ABS	50
Abrazadera	2	ABS o PLA	32
Escuadra trasera	4	ABS	36
Bisagra	2	Filamento flexible	13

La configuración recomendad para la impresión de cada pieza podrá encontrarse en el apartado 3.1.4. Para la impresión de los 3 primeros tipos de piezas se recomienda introducir una rotación de 180º con respecto al eje Y para un correcto acabado. Por otra parte, si no se dispone de filamento flexible, es posible encontrar multitud de diseños de bisagras sólidas para imprimir con cualquier otro material en internet en páginas como Thingiverse. Por otra parte para ahorrar tiempo y plástico también se adjunta a este trabajo una versión más pequeña de la escuadra trasera que también puede ser útil.

Una vez que se tienen las planchas recortadas, las piezas impresas, los listones de madera y los tornillos requeridos, podemos pasar al ensamblaje. Antes de continuar, nótese que los listones irán sobre el panel superior, cada uno centrado sobre los cuatro boquetes que hay a cada lado, por lo tanto deberán realizarse los mismos agujeros, comenzando por un extremo dejando un pequeño sobresaliente de 2 cm por la parte posterior.

En primer lugar habrá que atornillar sendas pinzas del marco a la plancha superior junto con los listones y las abrazaderas tal y como se puede observar en la siguiente figura:

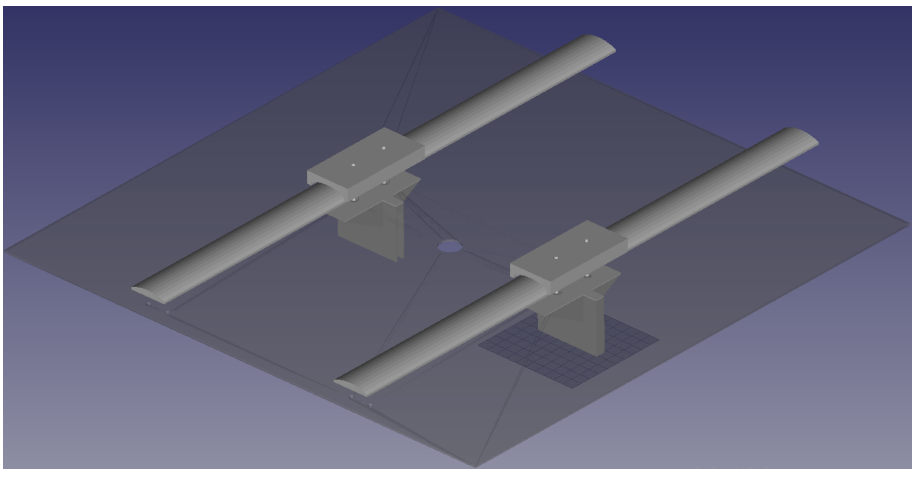


Figura 4.1 Montaje de la cara superior del cerramiento.

Una vez hecho esto podemos pasar a atornillar la plancha delantera a las bisagras, y estas al extremo delantero de los listones. A continuación, se unen las planchas laterales junto a la trasera con las 4 escuadras, una en cada esquina como se observa en la figura 4.2.

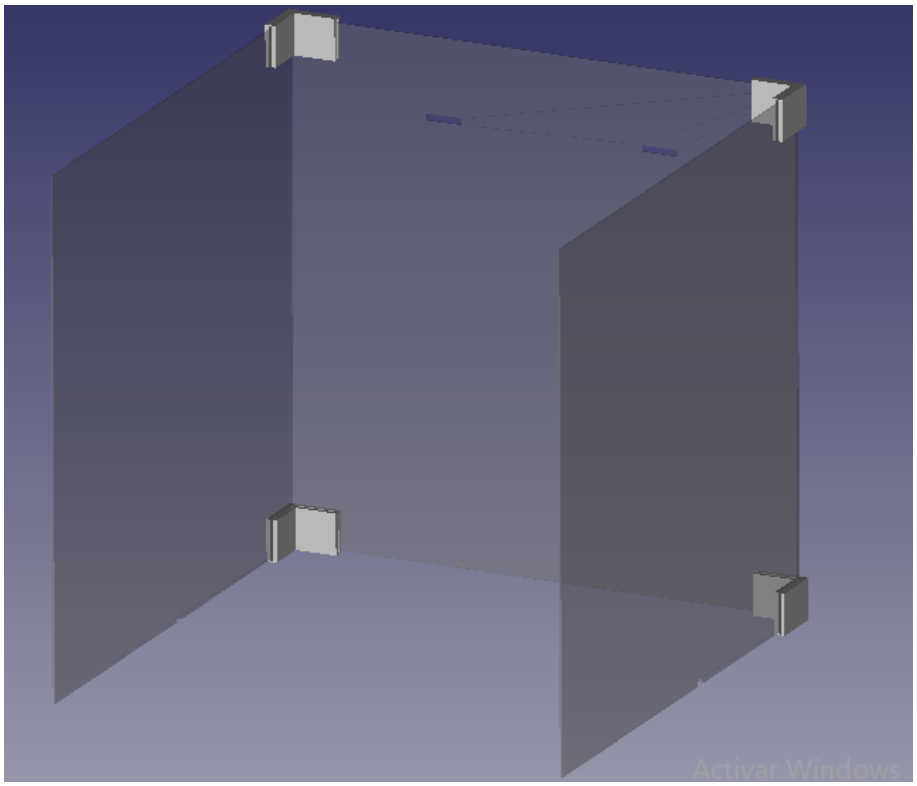


Figura 4.2 Montaje de la parte lateral y posterior.

Con todo montado, queda poner la parte superior y delantera sobre el marco de la impresora y colocar alrededor la parte posterior y trasera, y por último, asegurarse de que el sobrante de los listones de madera encaja en los orificios de la parte posterior, consiguiendo que las pinzas tengan una carga equilibrada y no venzan hacia adelante con el tiempo.

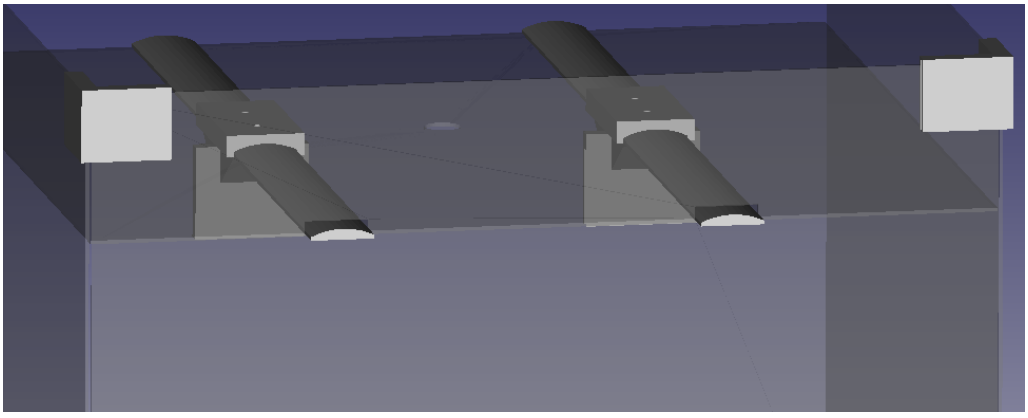


Figura 4.3 Detalle de la parte posterior.

4.1.3 Resultados

Tras tener el cerramiento completamente montado, el siguiente paso fue comprobar los resultados obtenidos con la impresora aislada. Para ello comencé realizar pequeñas impresiones de 30 - 40 minutos de duración con ABS. Durante estas impresiones pude observar resultados directos. En primer lugar, pude permanecer durante todo el trascurso en la misma habitación en la que se alojaba la impresora teniendo tanto la ventana como la puerta abierta y un ventilador de techo encendido, todo esto sin llegar a tener problemas de *warping*, el cual sí que tenía anteriormente con todo cerrado y el ventilador apagado. Por otra parte, el característico olor que desprende el ABS y el cual implica humos potencialmente cancerígenos desaparecieron hasta que

finalmente el panel delantero fue abierto.

Estos primeros resultados resultan satisfactorios, pero no reflejan todas las repercusiones que tiene el cerramiento de la impresora. Por ello se seleccionó una impresión de mayor tamaño con una duración total de 4 horas y 30 minutos aproximadamente. Para ella, se utilizó la siguiente configuración del *slicer*:

Tabla 4.2 Configuración para la impresión de prueba con ABS.

Parámetro	Unidad	Valor
Temperatura de la cama	ºC	110
Temperatura de la boquilla	ºC	245
Velocidad de impresión	mm/s	70
Velocidad de perímetro exterior	mm/s	60
Velocidad de relleno	mm/s	110
Densidad de relleno	%	20

Antes de comenzar con la impresión, se colocaron 4 sensores de temperatura LM35DZ repartidos por el interior de la impresora. Un primer sensor de temperatura colocado sobre la cama caliente a la altura de la pantalla LCD que se encuentra en la parte superior del marco. Gracias a éste una idea de la temperatura del aire del interior. Los otros 3 sensores se pegaron con cinta adhesiva kapton, la cual tiene grandes propiedades térmicas, en los motores del eje Y, del eje X y del eje del extrusor. Los otros 2 motores restantes, los del eje Z, no se tuvieron en cuenta por ser estos los que menos trabajo realizan y por tanto se calientan menos además de por estar en la zona más baja. Para realizar las medidas se hará uso del MSP430G2553, utilizando el conversor ADC10 de 10 bits de precisión del que dispone. El esquema de conexionado se puede observar en la figura 4.4 y el código utilizado se encuentra explicado en el siguiente capitulo.

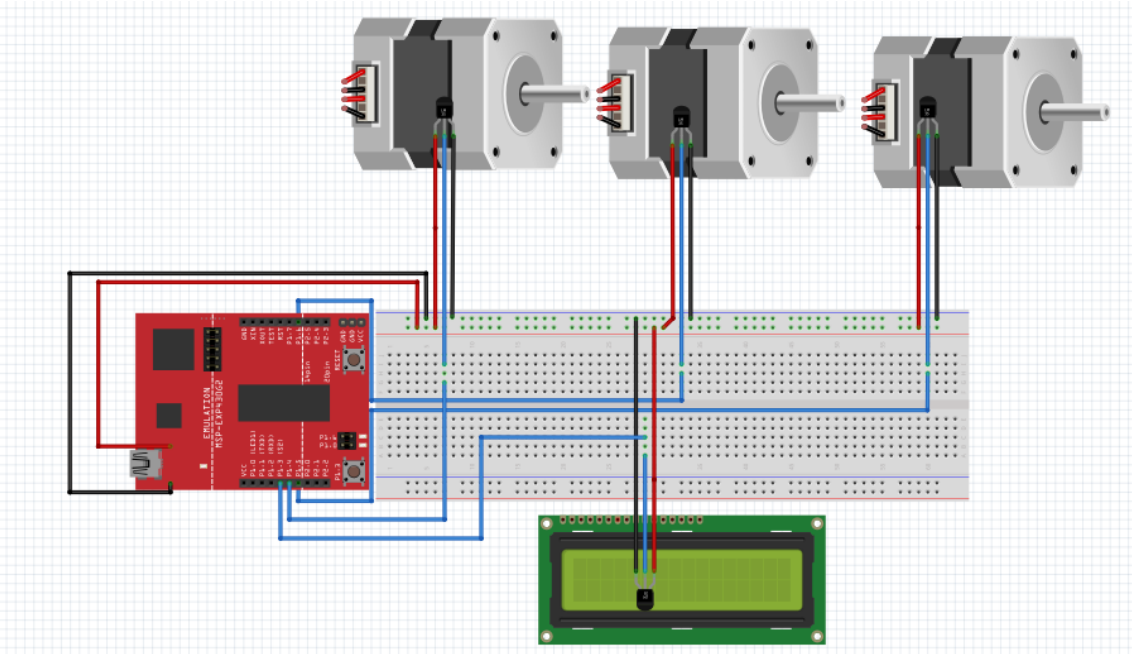


Figura 4.4 Cableado para la medición de temperaturas.

Cabe destacar en este esquema que la alimentación de los sensores de temperatura no se realiza mediante

VCC. Esto es debido a que el sensor LM35DZ necesita como mínimo 4V para asegurar el correcto funcionamiento del mismo, mientras que el microcontrolador ofrece 3.3V en el terminal VCC. Para ello se soldarán un par de conectores bajo la entrada USB del LaunchPad. De este modo, se aprovechará la alimentación de 5V proveniente del puerto USB que es rutado hacia los terminales TP1 y TP3 (ver figura 4.5). Esta técnica se utilizará tanto aquí como en los siguientes apartados. También es importante mencionar que a pesar de alimentar a 5 voltios los sensores de temperatura, estos no comprometerán la integridad del conversor ADC, dado que al tener una sensibilidad de 10mV por ºC y ser capaz de medir hasta 100ºC, la salida máxima del LM35DZ será de 1 voltio.

Figura 4.5 Terminales a 5V del LaunchPad.

Una vez transcurrida la impresión se procede a hacer un estudio de los resultados de las temperaturas obtenidas. Cabe destacar que las mediciones obtenidas de los sensores de temperatura son muy oscilatorias, sobre todo los sensores colocados en los motores. Esto es debido a la contaminación electromagnética que proviene de los motores paso a paso. Pero a pesar de ello, es posible sacar conclusiones bastante claras sobre las temperaturas registradas.

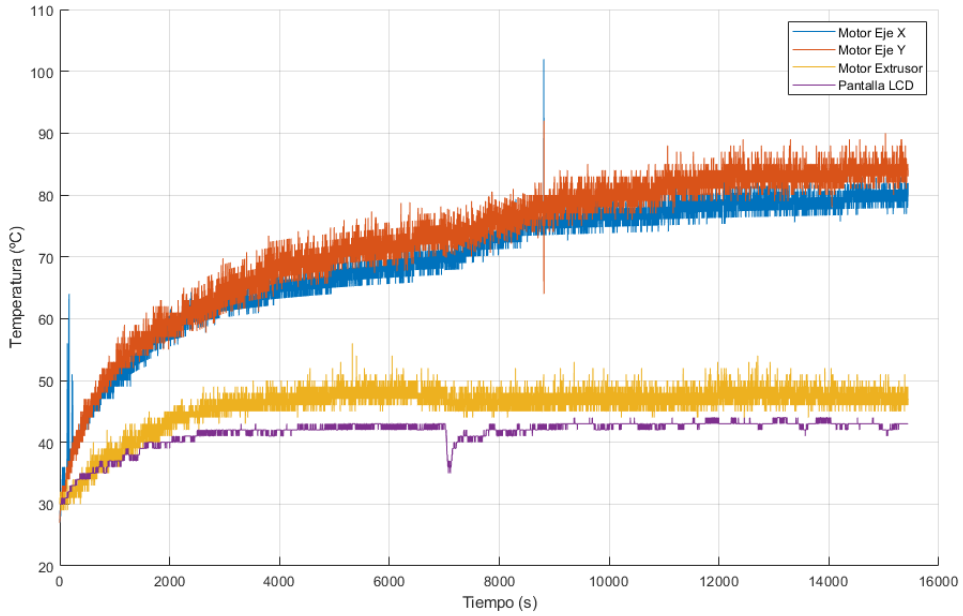


Figura 4.6 Gráfica de temperaturas de los motores y la pantalla.

Se puede observar a simple vista como los motores más comprometidos son los del eje X e Y, los cuales transcurridas 3 horas (10800 s) se encuentran entorno a los 80ºC. Ésto es debido al autocalentamiento que

sufren al estar continuamente en movimiento y la falta de refrigeración. Esto supone un potencial problema, ya que en impresiones más largas, lo cual es frecuente en piezas de dimensiones medias, puede producir una degradación de los motores debido a la desmagnetización de estos, así como una degeneración de las correas cuyo limite de operación se encuentra entorno a los 80ºC.

Por otra parte, se puede observar como la temperatura del extrusor, el cual opera a mucho menor velocidad, se mantiene estable entorno a los 48ºC y la pantalla LCD sobre los 43ºC. Esta última temperatura nos da una idea de la temperatura a la que se estabiliza el aire cerramiento, la cual se mantiene en unos límites fuera de riesgos.

Otro aspecto que puede llamar la atención son los picos de temperatura que se registran en los motores X e Y, los cuales no significan nada, ya que son medidas erróneas puntuales del los sensores.

En conclusión tras las pruebas realizadas, los resultados de la impresora se ven mejorados, evitandose problemas anteriormente frecuentes (*warping*, taponamiento de la boquilla...) y haciendo más segura la convivencia con ésta máquina. Por otro lado, se ve la necesidad de introducir elementos de refrigeración para los motores mencionados, los cuales en impresiones de una duración superior a 4 horas, estarán operando a temperaturas nada recomendables, que a la larga acabaran provocando fallos que pueden inhabilitar la impresora.

4.2 Pruebas de refrigeración

Como se mencionó en el apartado 3 la solución por la que se va a optar a la hora de la refrigeración serán las celdas peltier y un modulo L298N para alimentar estas celdas y regular su temperatura. Ambos elementos deberán ser alimentados a 12V, que es justo la tensión que nos ofrece la fuente de alimentación de la propia impresora. Esta elección se ha hecho para poder realizar un cierto control sobre la temperatura de los motores, de forma que estos se encuentren a una temperatura de operación aceptable, y que el consumo de las celdas peltier sea el mínimo que asegure estas condiciones. Antes de pasar a la refrigeración de los motores, se harán ciertas pruebas sobre el funcionamiento de estas celdas para comprobar su funcionamiento.

4.2.1 Peltier sin aletas

En primer lugar se realizara el experimento de alimentar la celda peltier a diferentes puntos de operación, realizando un control PWM sobre el L298N con la celda peltier conectada a la salida de este. La señal PWM se generará con uno de los timers que posee el MSP430, rutado hacia las salidas 2.1 y 2.2, con un periodo de 62,5 microsegundos, de forma que el efecto sobre la salida del L298N sea prácticamente continua. El duty cycle se irá incrementando desde el 0 % hasta el 100 % de 20 en 20. El esquema de conexionado será el siguiente:

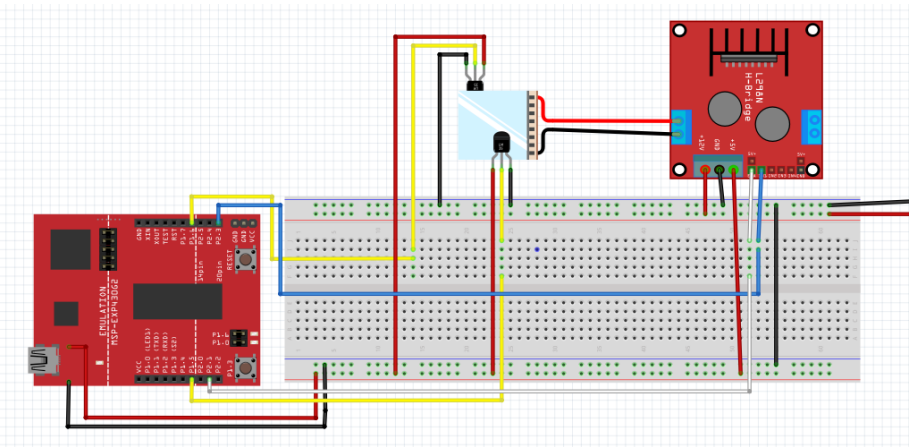


Figura 4.7 Esquema de conexionado para pruebas con la celda peltier.

Los cables rojo y negro que aparecen sin conectar irán directamente conectados a la fuente de alimentación de la impresora a los terminales de 12V y GND respectivamente. El programa hecho para esta prueba se puede encontrar en el apendice B. Los resultados obtenidos en esta primera prueba en la que la peltier se encuentra con sendas caras descubiertas quedan reflejados en la gráfica de la figura 4.8, en la que se puede observar como se crea una diferencia de temperatura entre ambas caras que aumenta proporcionalmente con el duty cycle subministrado en el enable del L298N (cable blanco), llegando a una diferencia de 26ºC (89ºC-63ºC). Pero esta diferencia de temperatura se hace en base a temperaturas nada deseables para la refrigeración, puesto que aumenta constantemente en ambas caras.

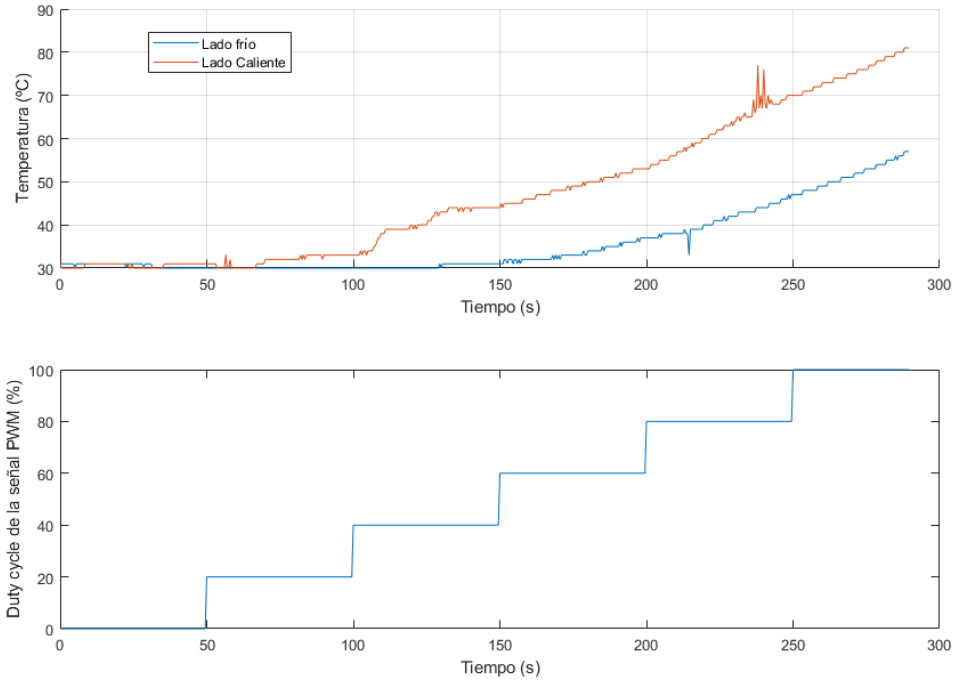


Figura 4.8 Gráfica de temperaturas de ambas caras de la celda peltier sin aletas.

Es por esto que el conjunto de aletas y ventilador es muy necesario si lo que se desea es crear una diferencia de temperatura que disminuya con respecto a la temperatura ambiente de la que se parte.

4.2.2 Peltier con aletas

En este caso se pasa a analizar los resultados de realizar el mismo experimento que antes, sin variar el cableado, incluyendo únicamente un pequeño ventilador y un disipador en la cara caliente de la celda peltier. Se conectará el ventilador a los 12 voltios, ya que es ésta su tensión nominal, y se colocara de forma que el flujo de aire este orientado hacia las aletas del disipador. Se vuelve a arrancar el mismo programa que para el caso anterior obteniendo en este caso los resultados reflejados en la figura 4.9. En dicha gráfica se puede observar como el lado caliente se mantiene fijo a una temperatura ligeramente mayor que la ambiente, y toda la potencia invertida en crear una variación de temperatura se invierte en disminuir la temperatura de la cara fría por debajo de los 10ºC. Se alcanza el mismo salto de temperatura que en visto en el apartado anterior, pero en este caso el salto de temperatura de 26ºC se hace desde los 35ºC a los 9ºC.

Teniendo en cuenta que el encapsulado de la celda peltier es de óxido de aluminio o alumina (Al_2O_3), material que posee una conductividad térmica de alrededor de 30W/m*K, sabiendo que las dimensiones de la celda peltier son de 40x40x3,8mm, si suponemos que la celda esta hecha entera de alumina, se puede calcular la transferencia de calor que lograremos con la celda ante los distintos duty cycles aplicados según la

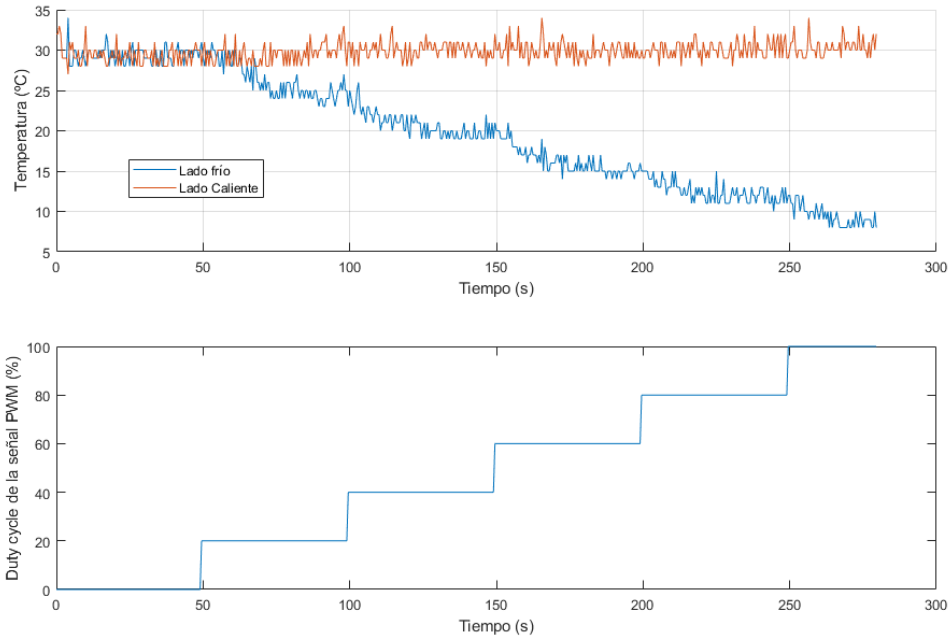


Figura 4.9 Gráfica de temperaturas del disipador y de la cara fría de la celda peltier.

formula:

$$\frac{Q}{t} = \frac{kA \triangle T}{d} \tag{4.1}$$

Donde $\frac{Q}{t}$ es la potencia calorífica transmitida, k es la conductividad térmica, A es el área, d la profundidad y $\triangle T$ es el incremento de temperatura. Realizando el cálculo para los datos registrados obtenemos los resultados mostrados en las gráficas de la figura 4.10.

4.2.3 Conclusiones

Tras las diferentes pruebas realizadas, y teniendo en cuenta el rango de temperaturas admisibles para las celdas peltier indicadas en la tabla 3.5, es evidente la necesidad de incluir el disipador y el ventilador junto a ellas. En primer lugar para mantener las celdas dentro del rango de temperaturas admisibles, y en segundo lugar para conseguir disminuir la temperatura de la unión entre las celdas peltier y el motor.

Por otra parte, con los datos registrados se podrá hacer un primer ajuste del control a realizar sobre dichas celdas a la hora de introducirlas en los motores de la impresora, aunque este ajuste pueda variar posteriormente, ya que los cálculos realizados están ligados a ciertas suposiciones.

4.3 Montaje del sistema de control

Una vez detectadas las necesidades y una posible solución para afrontarlas será necesario diseñar un par de soportes que permitan fijar el conjunto de celda peltier, disipador y ventilador a los motores paso a paso, de forma que se asegure el mayor contacto posible en todo momento entre estos elementos. Además habrá que realizar el cableado de forma que se aseguren todas las conexiones sin que los elementos móviles de la impresora interfieran en éstos.

4.3.1 Diseño

Para el diseño de los soportes para los motores, el primer paso ha sido tomar medidas de los motores paso a paso del eje X e Y para poder modelarlo en Freecad. Éstos tienen unas dimensiones aproximadas de 42x42x34mm. Por otra parte, se puede observar que en el modelo específico para el que se realiza este diseño, ambos motores que se quieren tratar tienen disponibles uno de los 4 orificios disponibles para atornillar

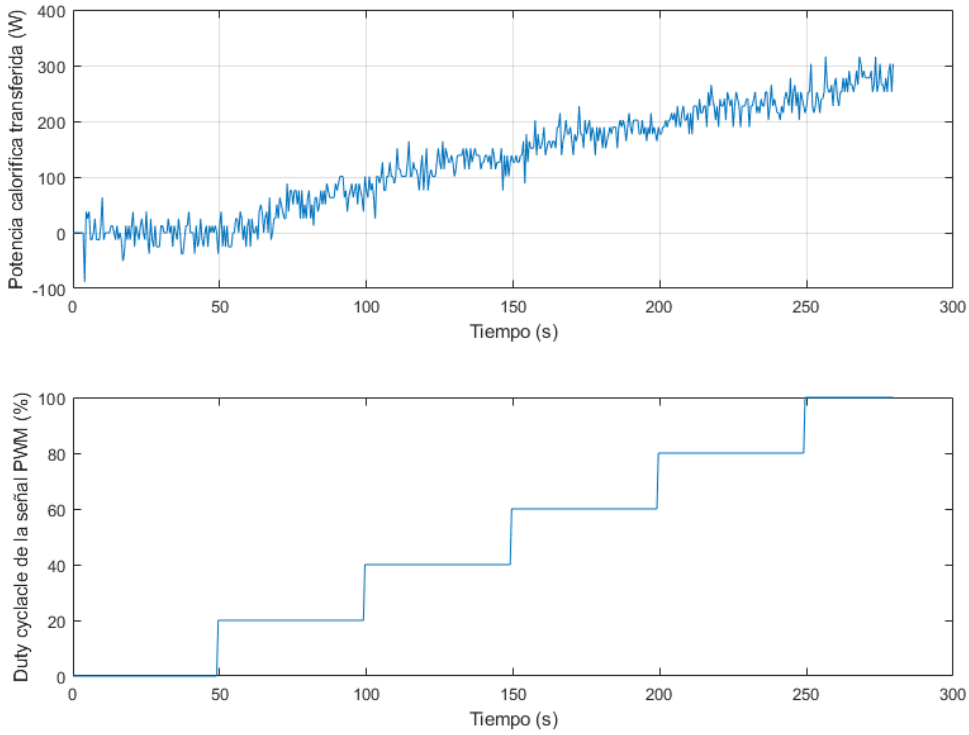


Figura 4.10 Potencia calorífica instantánea transmitida por la celda peltier estimada .

los motores. Con todo esto, la solución propuesta para los soportes será una pieza que se apoye en la parte superior de los motores, con un pequeño saliente con un orificio para fijar con un tornillo el soporte a los motores.

Además, estos soportes deberán disponer de un espacio en el que poder encajar el conjunto de ventilador, disipador y celda peltier, así como para sujetar los sensores de temperatura de forma que entren en contacto con la armadura del motor. Para ello se ha diseñado, por una parte, que las piezas tengan un pequeño apartado en la parte posterior del motor de 43x43x21mm, correspondiente a las dimensiones de ancho y alto de las celdas peltier y del disipador, y una profundidad correspondiente a la suma de las profundidades de los tres elementos. Por otra parte, en uno de los laterales se habrá habilitado un orificio rectangular de 9,5x3 mm en los que poder encajar los cables que sujetarán el sensor LM35DZ. También se colocará una pequeña pletina que asegure que dicho sensor esté en contacto con la carcasa del motor.

Como se puede observar en la figura 4.11, el espacio disponible para los elementos de refrigeración tendrá un espacio que permita la entrada de aire para el ventilador en la parte exterior, y la parte interior comunica directamente con la parte trasera del motor de forma que la celda peltier esté en pleno contacto con éste.

Por otro lado, como se menciono en el capítulo anterior el motor situado bajo la cama caliente, es decir, el del eje Y, tiene ciertas restricciones espaciales, ya que se encuentra enfrentado a los raíles sobre los que se apoya dicha cama, y por tanto necesitará ciertas modificaciones con respecto al soporte de la figura 4.10, ya que si se pusiera este soporte en este sitio, se verían limitados los movimientos de a cama caliente. Para evitar esto se han realizado diferent es cortes a la pieza del soporte del eje X de forma que queden libre todos los movimientos relacionado a la cama caliente. El resultado tras los cortes se pueden observar en la figura 4.12.

Entre las modificaciones realizadas se puede observar un chaflán en la parte superior trasera, el cual permite que la cama caliente se pueda mover libremente, y una sección en la esquina inferior el cual evita que el soporte choque con una tuerca de las barras de la propia estructura de la impresora.

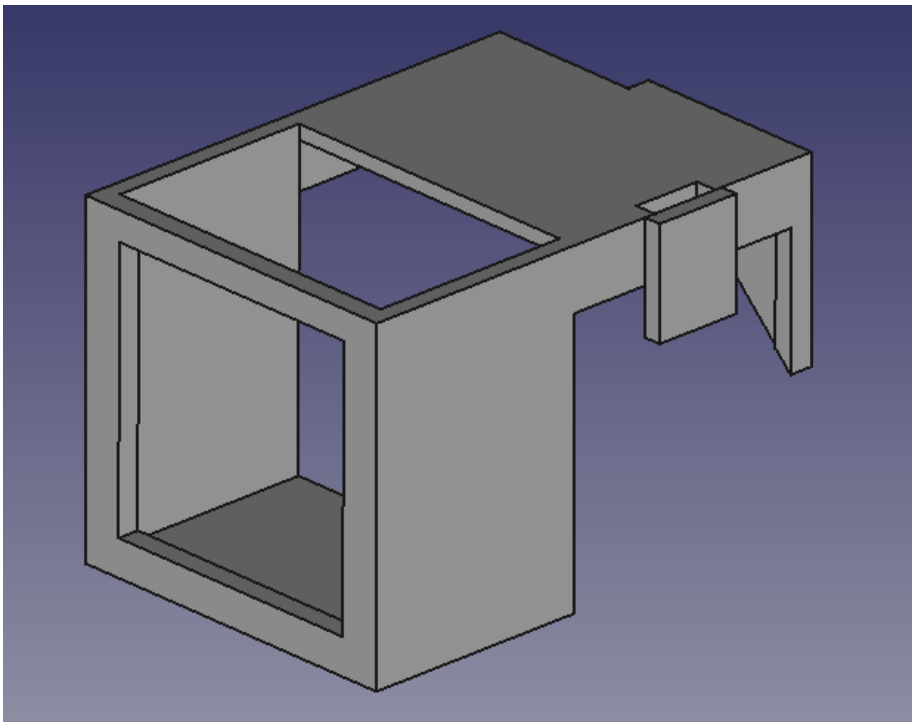


Figura 4.11 Detalle del soporte del motor paso a paso del eje X.

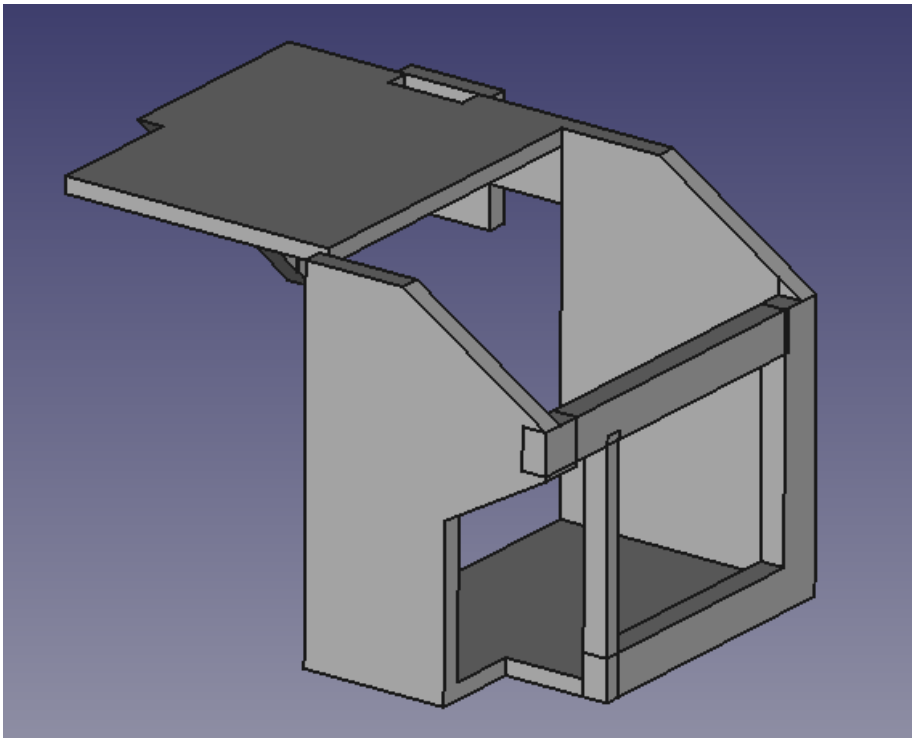


Figura 4.12 Detalle de las secciones realizadas al soporte del eje Y.

4.3.2 Montaje

Una vez que se tienen sendos diseños necesarios para afianzar los elementos de refrigeración a los motores, habrá que montarlos junto al resto de elementos necesarios para el control, como el módulo L298N, de forma que se permitan todos los movimientos normales de la impresora. Para ello se utilizarán los cables de conexión y 2 mini protoboards.

El esquema de conexionado es el mostrado en la figura 4.13. En él cabe destacar que las protoboards que se pueden observar en el croquis no se corresponden con las del montaje físico. De hecho con la mini protobard de la izquierda se pretende representar la fuente de alimentación de la impresora.

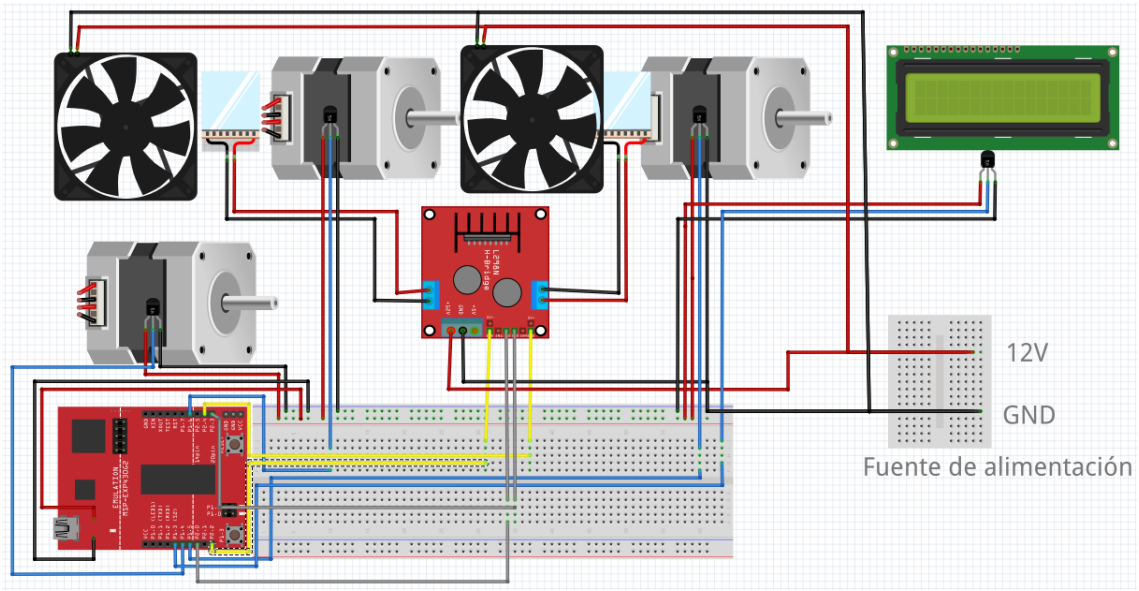


Figura 4.13 Esquema de conexionado del sistema de control.

En cuanto al control realizado para asegurar que la temperatura de los motores se mantenga en todo momento en unos límites estables, dado que los sistemas de transferencia de calor son sistemas muy lentos, se ha optado por una estrategia sencilla de comparación con una serie de condicionales encadenados, en los cuales en función de la temperatura leída en los motores, se asignará un duty cycle a las señales PWM que controlan las celdas peltier. De esta manera, se definirán tramos de temperatura en los que las celdas deberán funcionar a distintos rendimientos constantes, asegurando una transferencia de calor forzada y constante de los motores hacia las aletas. Los tramos definidos para ambos motores serán 60ºC-69ºC, 70ºC-79ºC, 80ºC-89 y de 90º en adelante, teniendo asociados respectivamente los siguientes duty cycle: 25 %, 50 %, 75 % y 100 %. De esta forma se tratará mantener estable la temperatura, sin llegar a tener funcionando las celdas peltier a pleno rendimiento, tratando de reducir el consumo de las mismas.

5 Programación del MSP430G2553

Tanto para realizar las pruebas como para el programa en sí del control de temperatura será necesario programar el microcontrolador MSP430G2553. Éste es un microcontrolador bastante versátil, el cual se programa en C, y que ofrece varios recursos que serán necesarios para este proyecto. Es importante la comprensión a bajo nivel de los distintos elementos para su correcta utilización. A lo largo de este capitulo se hará una descripción de los diferentes elementos del microcontrolador que se van a usar, así como el código empleado para su uso.

Para comprender debidamente el funcionamiento y la programación de este microcontrolador hay que comenzar por comprender su arquitectura. Al igual que el resto de micros de la familia MSP430, este microcontrolador tiene una CPU de tipo RISC de 16 bits, lo cual significa que tiene un conjunto de instrucciones reducidas. Esto implica que, a diferencia de otro microcontroladores, este tiene una serie de instrucciones básicas que entiende el procesador y por lo tanto para realizar operaciones complejas, habrá que descomponer éstas en instrucciones en una serie de instrucciones sencillas que serán ejecutadas sucesivamente. Esta arquitectura está enfocada a un hardware sencillo, y por tanto más económico, dejando la complejidad para la parte de software. Por otra parte, el micro tiene una memoria no volátil de tipo Flash de 16 KB y una memoria RAM de 512 B. Al ser bastante limitada la memoria hace que haya que poner especial ojo a la hora de su programación en aspectos como la elección del tipado de las diferentes variables, ya que se puede agotar la memoria fácilmente.

Un aspecto muy interesante de esta familia de microcontroladores es que posee diferentes modos de bajo consumo, con los cuales se puede dejar a estos en un estado de reposo en el cual se desactivan diferentes características, logrando que el consumo sea prácticamente despreciable. Esto es ideal para aplicaciones en los que se requiera alimentar el micro con baterías.

Este microcontrolador en particular tiene 2 puertos de 8 bits de entrada/salida, que a su vez y dependiendo del puerto, pueden ser rutados a otros elementos tales como el conversor AD o los temporizadores. Además, ambos puertos tienen la capacidad de hacer saltar interrupciones. Éstas permiten parar el código principal que este en ejecución o salir de los modos de bajo consumo para saltar a la rutina indicada justo en el momento de la activación de la interrupción, volviendo a la ejecución del código principal una vez acabada la rutina. En la figura 5.1 se pueden observar ambos puertos y todas sus características asociadas.

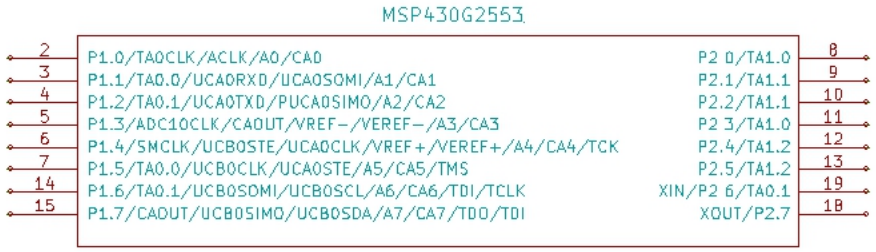


Figura 5.1 Pines del MSP430G2553 con sus características.

El método de configuración de los diferentes periféricos y puertos del MSP430 es mediante la inicialización de diferentes registros. De esta forma cada recurso tendrá diferentes registros de un byte asociados, de los cuales cada bit o conjuntos de bits indicaran diferentes opciones. A la hora de configurar estos registros, en cuanto a código se refiere, se hará uso constantemente de máscaras. Estas máscaras son variables de valor constantes, las cuales representaran un valor en binario correspondiente con los bits necesarios para activar diferentes opciones, las cuales aportan facilidades tanto para escribir como para leer código. Por otro lado, se pueden distinguir principalmente 4 operaciones booleanas básicas:

Tabla 5.1 Operadores booleanos básicos de C.

Operación booleana	Sintaxis	Uso común	Ejemplo
NOT	$\sim$	Deshabilitar alguna configuración previamente habilitada	$ADC10CTL0\ \&= \sim ENC$
AND	&	Seleccionar los una serie de bits. Equivale a una multiplicación	$IFG2\&UCA0RXIFG$
OR	|	Añadir bits. Equivale a una suma	$P1DIR = P1DIR\|BIT0$
XOR	$\wedge$	Alternar el valor de un bit	$P2OUT = P2OUT\hat{}BIT6$

Es importante destacar que estos operadores se pueden acortar a la hora de hacer una asignación, de forma que en vez de escribir una asignación como en el código de ejemplo para el operador OR, se podría escribir $P1DIR| = BIT0$. A continuación se pasa a mostrar los registros asociados a los recursos utilizados durante el proyecto, así como el código utilizado para su configuración.

5.1 Pines E/S

Los pines de entrada/salida, también conocidos como GPIO (General Purpose Input Output), del microcontrolador son la interfaz física entre el programa escrito y cualquier periférico que queramos comunicar con éste como por ejemplo sensores u otros microcontroladores. Los registros que se describen a continuación estarán duplicados para el puerto 1 y 2, de forma que para hacer referencia a cada puerto habrá que sustituir la x por el número del puerto deseado. Dichos registros de configuración de los pines de entrada salida son:

5.1.1 PxSEL y PxSEL2

Ambos registros son de escritura y sirven para elegir a que recurso queremos rutar los diferentes pines. Como algunos pines disponen de hasta 4 posibles funciones, se necesitan 2 bits para seleccionar la deseada. Cada uno de los 8 bits de ambos registros está ligado a cada uno de los 8 pines de cada puerto. El número de selección dependerá del recurso que se quiera seleccionar y del puerto, siendo el 00 utilizar los pines como E/S digital salvo para los pines 6 y 7 del puerto 2 que con el código 00 servirán de entrada para un reloj externo. Ambos registros se inicializan a 0 por defecto.

5.1.2 PxDIR

Este registro servirá para seleccionar la dirección de los puertos configurados como E/S siendo el valor 0 Entrada y el valor 1 salida. Este registro es de escritura y se inicializa con todos los pines como entradas.

5.1.3 PxOUT

Registro de escritura que recoge los valores que se escribirán sobre los pines previamente seleccionados como pines de salida. El valor 1 corresponde con la salida a tensión Vcc, y el valor 0 con GND. Todos los

pines comienzan por defecto a 0.

5.1.4 PxIN

Este es un registro de solo lectura en el cual quedarán reflejadas las lecturas de los pines que hayan sido seleccionadas como entrada, de forma que valdrán 1 si registra un nivel alto de tensión ($\approx Vcc$) o 0 si queda registrado un valor negativo de tensión ($\approx GND$) .

5.1.5 PxREN

Con este registro de escritura se puede seleccionar si habilitar resistencias de Pullup o Pulldown en los pines seleccionados como entradas. Para habilitarlas habrá que asignar un 1 al bit asociado al pin deseado. Para seleccionar si se desea una resistencia de Pullup o Pulldown habrá que escribir en un 1 o un 0 respectivamente en el registro PxOUT.

5.2 Módulo de selección de reloj

Con el MSP430 podremos seleccionar varias velocidades de reloj de diferentes fuentes, siendo posible seleccionar una frecuencia máxima de 16MHz. Existe también la posibilidad de introducir un reloj externo, teniendo para ello habilitado los pines 6 y 7 del puerto 2. Todo esto será manejado con el módulo básico de reloj. Las tres señales de reloj disponibles son: ACLK (reloj auxiliar), MCLK (reloj principal, usado por la CPU), SMCLK (reloj sub-principal). Todos ellos podrán ser divididos por un factor de 1, 2, 4 u 8. Los registros utilizados para su configuración se describen a continuación.

5.2.1 DCOCTL

Este registro configura el oscilador controlado digitalmente (DCO), siendo posible elegir entre 1MHz, 8MHz, 12MHz y 16MHz.

5.2.2 BCSCTL1

En este registro se podrá seleccionar si se desea el divisor para el ACLK de dividendo 1, 2, 4 u 8 mediante **DIVAx** (x=0...3).

5.2.3 BCSCTL2

En este registro se seleccionarán loso osciladores deseados para MCLK y SMCLK mediante **SELMx** y **SELSx**. Para el primero existen 3 opciones, siendo el DCOCLK equivalente a los indices 0 y 1. Para el segundo reloj tendremos solo dos posibilidades, siendo el indice 0 reservado para el DCOCLK nuevamente.

A su vez se podrán seleccionar los divisores para ambos relojes con **DIVMx** y **DIVSx** de la misma forma que se selecciona para el reloj ACLK.

5.3 Temporizador

Otro de los módulos integrados del MSP430 es el temporizador Timer A, el cual es un contador/temporizador de 16 bits con tres registros de comparación. El temporizador permite múltiples modos de captura/comparación, interrupciones temporizadas e incluso generar señales PWM. En nuestro caso este módulo sera útil para simular una señal analógica mediante modulación de ancho de pulsos (PWM).

En los siguientes subapartados se describen los registros empleados para generar señales PWM.

5.3.1 TACTL

El registro de control del timer A se puede seleccionar en primer lugar el reloj que alimente al temporizador mediante **TASSELx**, estando disponibles desde 0 a 2 TACKL, ACLK y SMCLK. También se podra elegir uno de los 4 divisores disponibles (1, 2, 4 u 8) con **IDx**. Con la máscara **MCx** podemos seleccionar el modo de funcionamiento del timer, siendo estos modos de 0 a 3: timer en pausa, modo ascendente hasta la cuenta **TACCR0**, modo continuo (cuenta desde 0 hasta 0FFFFh), o modo ascendente/descendente recorriendo desde 0 hasta **TACCR0** y volviendo a 0.

Es aquí donde se pueden habilitar también las interrupciones mediante **TAIE** o resetear el contador mediante **TACLR**.

5.3.2 TACCRx

En estos registros se podrá poner el limite superior del contador si se esta actuando en modo comparación, o si no, será donde se almacene el valor de al cuenta cuando se haga una captura si se está en este modo.

Cabe destacar que para generar las señales PWM se hará uso del registro TACCR0 para marcar el periodo de la señal ya que será el que marque el fin de la cuenta del contador de 16 bits, y se hará uso de los otros dos registros disponibles TACCR1 y TACCR2 para designar el duty cycle de las diferentes señales PWM.

5.3.3 TACCTLx

Tanto este registro como el anterior estará triplicado para cada una de los 3 canales de los que dispone el timer A (0, 1 y 2), permitiendo con el mismo módulo emplear el timer de distinta forma.

En este registro seleccionaremos mediante **CMx** si se desea utilizar el correspondiente canal en modo captura o no (>0 o 0), y en caso de estar en modo captura, podemos seleccionar si se desea activar la captura por flanco de subida, bajada o ambos (1, 2, 3). Para emplear el timer como generador de señales PWM se seleccionara esta máscara con indice 0. Además, se podrá seleccionar la acción que este timer tenga sobre la salida en caso de haberse rutado hacia alguno de los pines del GPIO gracias a **OUTMODx**. Los diferentes modos que se pueden seleccionar sobre la salida son:

- 0 - El valor del bit OUT.
- 1 - Set.
- 2 - Toggle/reset.
- 3 - Set/reset.
- 4 - Toggle.
- 5 - Reset.
- 6 - Toggle/set.
- 7 - Reset/set.

5.4 Conversor AD

Este microcontrolador tiene un modulo de conversión analógico-digital de 10 bits de precisión bajo el nombre de ADC10, el cual implementa tanto el conversor AD de aproximaciones sucesivas con hasta 8 canales de entrada, así como un selector de fuente de tiempo de muestreo, un generador de referencias capaz de elegir entre varias referencias internas o referencias externas y un controlador de transferencia de datos.

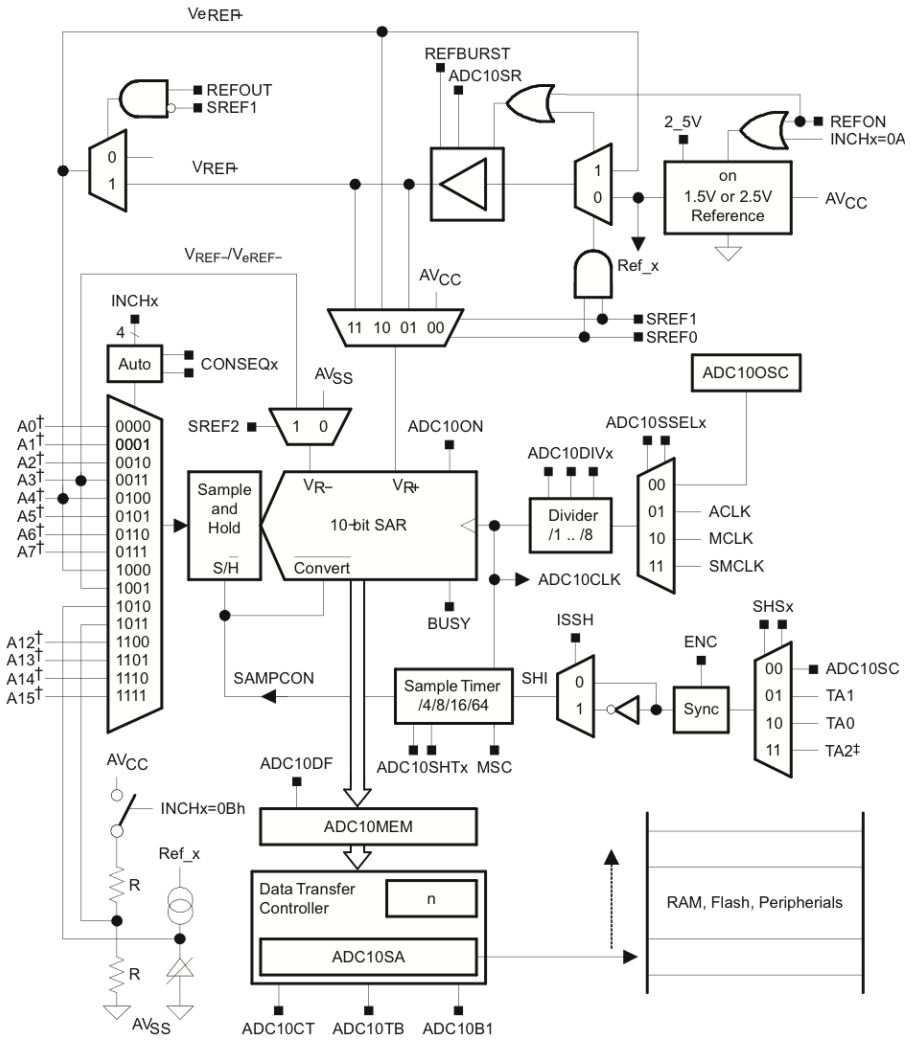


Figura 5.2 Esquema del módulo ADC10.

5.4.1 ADC10CTL0

Este registro de lectura/escritura de 16 bits es el relativo a gran parte de las configuraciones de las que dispone este conversor. Entre las configuraciones más interesantes de este módulo se encuentran la selección de la referencia de la conversión. Se podrá elegir el modo deseado gracias a la mascara **SREFx** donde x es el número del modo, estando disponibles los siguientes modos:

- 0 - V+= Vcc y V-=GND.
- 1 - V+= Vref+ y V-=GND.
- 2 - V+= Veref+ y V-=GND.
- 3 - V+= Veref+ y V-=GND, siendo el valor de Veref+ guardada previamente en un buffer.
- 4 - V+= Vcc y V-=Veref-.
- 5 - V+= Vref+ y V-=Veref-.
- 6 - V+= Veref+ y V-=Veref-.
- 7 - V+= Veref+ y V-=Veref-, siendo el valor de Veref+ guardada previamente en un buffer.

También podemos encontrar la máscara **ADC10SHTx** con la cual podremos seleccionar el preescalado de reloj que alimentará al circuito de muestreo y retención, siendo desde 0 a 3 los valores de escalado x4, x8, x16 y x64.

Si se desea hacer más de una conversión se debe habilitar el bit **MSC**, el cual permite configurar los modos de conversión en secuencia y conversiones repetidas que se verán a continuación. Será obligatorio, si se desea utilizar el conversor, activarlo con la máscara **ADC10ON**. Todas estas configuraciones han de hacerse con el bit **ENC** a 0, el cual hace referencia a la habilitación de la conversión, de modo que no podemos configurar el modulo ADC10 mientras se está utilizando. Junto con esta ultima opción se suele habilitar la máscara **ADC10SC** para comenzar la conversión, si se desea lanzar "a mano" en vez de mediante algún temporizador.

Otras configuraciones interesantes se pueden activar gracias a este registro como por ejemplo habilitar interrupciones cuando el conversor haya acabado mediante **ADC10IE**, aunque no serán usadas en los códigos realizados.

5.4.2 ADC10CTL1

Éste es otro registro de lectura/escritura dedicado a la configuración de este módulo. En él configuraremos el canal por el que se desee comenzar la conversión con **INCHx** donde x recorre desde 0 a 11, siendo los 8 primeros valores los referentes a los 8 canales de entrada asociados a los pines A0-A7, el valor 8 es para el valor de referencia externa Veref+, 9 para el valor de Vref/Veref-, 10 para la conversión de un sensor de temperatura que lleva integrado el propio microcontrolador, y el 11 para el valor medio de tensión de referencia del microcontrolador (Vcc-GND)/2.

Por otro lado encontramos **SHSx** para seleccionar el desencadenante de la conversión. Con el indice 0 se selecciónа el bit **ADC10SC** del registro visto anteriormente como desencadenante y con los indices 1,2 y 3 se seleccionan las salidas 0, 1 y 2 del Timer A respectivamente. Aquí también se podrá elegir un divisor del reloj de entrada del conversor, siendo posible seleccionar un dividendo de valor de 1 a 8 con la máscara **ADC10DIVx** (x=0...7). Así como el divisor, también se configura aquí el reloj de entrada que se desea mediante **ADC10SSELx** estando disponible, desde 0 a 3, el oscilador interno del conversor ADC10OSC, ACLK, MCLK y SMCLK. Por último encontramos la configuración del modo de conversión que se desea mediante **CONSEQx**. Los diferentes modos en los que puede operar éste son (de 0 a 3):

- Conversión de un solo canal.
- Conversión de una secuencia de canales.
- Conversión de un solo canal repetidamente.
- Conversión de una secuencia de canales repetidamente.

Por otro lado encontramos en este registro un bit de solo lectura llamado **ADC10BUSY** el cual nos indicará con 1 uno si se encuentra activo haciendo una conversión y con un 0 que ninguna operación se esta realizando.

5.4.3 ADC10AE0

En este registro de lectura/escritura se habilitarán los canales de entrada del conversor, de forma que se habilitarán poniendo a 1 los bits correspondientes a los puertos de los que se deseen hacer una lectura. Solo será necesario habilitar los canales correspondientes a los pines del GPIO.

5.4.4 ADC10MEM

En este registro de solo lectura se encuentran los 10 bits correspondientes a la última conversión realizada.

5.4.5 ADC10DTC0

Con este registro se podrá configurar el módulo de transferencia de datos, gracias al cual no será necesario estar continuamente haciendo lecturas del registro ADC10MEM. En el podemos configurar si se desea hacer uno o dos bloques de transferencia poniendo **ADC10TB** a 0 o a 1 respectivamente y si queremos hacer una sola transferencia (0) o transferencias continuamente (1) con **ADC10CT**.

5.4.6 ADC10DTC1

En este registro se deberá escribir el número de transferencias que se desea realizar en cada bloque. En el caso del trabajo, aunque queramos realizar solo 4 lecturas, se tendrá que indicar que haga la transferencia de los 7 canales (desde 0 a 6), y se despreciarán las correspondientes al los canales 0, 1 y 2. De no ser así las transferencias que realiza no se corresponderán con las deseadas.

5.4.7 ADC10SA

Este registro almacenará la dirección inicial en la que se realizará la transferencia de los datos del conversor. Por lo tanto se pasará un puntero a la variable en la que se desea guardar los datos.

5.5 Interfaz de comunicación serie universal

Otra de las grandes funcionalidades que nos ofrece este microcontrolador son dos interfaces preparadas para diversos modos de comunicación serie universales, la USCI_A y USCI_B. Ambas soportan diferentes tipos de comunicación. La primera se puede usar para comunicación UART, IrDA LIN y SPI; y la segunda se puede usar para I^2C y SPI.

Para este trabajo se utilizará el módulo USCI_A, más específicamente el módulo 0, ya que dispone de dos módulos USCI_A, en modo UART para podre transmitir los diferentes datos leídos al ordenador mediante el propio USB que trae integrado el LaunchPad. Un dato importante si se desea dicho puerto USB es la necesidad de conectar los jumpers referentes a las líneas RX y a TX del LaunchPad debidamente, siendo necesario colocarlos perpendicular al resto, tal y como se puede observar en la figura 5.3.

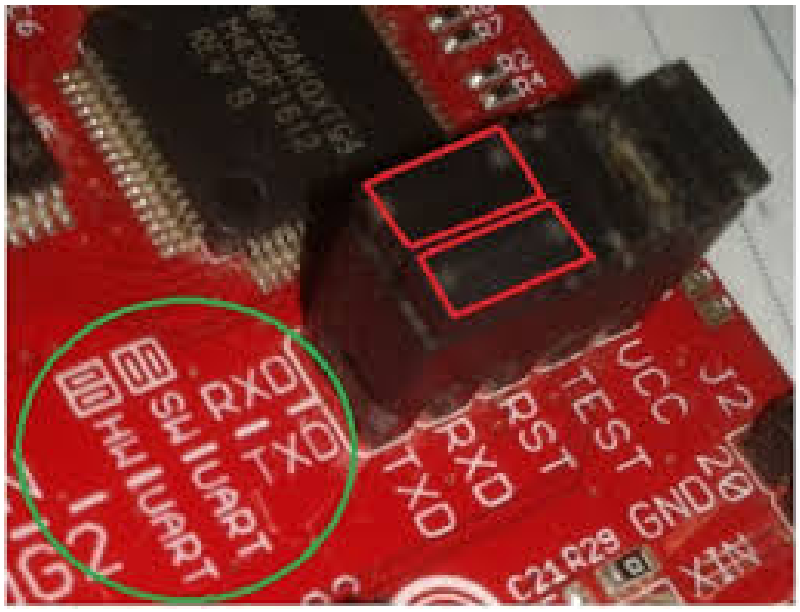


Figura 5.3 Conexionado de los jumpers del LaunchPad para el uso de la UART.

Tras este inciso se procede a describir las configuraciones necesarias para tener funcionando la UART, así como para configurar el formato de los mensajes transmitidos.

5.5.1 UCAxCTL0

Este registro, duplicado para USCI_A0 y USCI_A1, contiene diferentes configuraciones sobre el formato de la trama. Entre ellas encontramos la habilitación del bit de paridad con **UCPEN**, la selección de si la paridad es par (1) o impar (0) con **UCPAR**, si se desea empezar por el bit más significativo con **UCMSB** o con el menos significativo en su defecto, de si queremos utilizar 1 bit de stop o 2 asignando **UCSPB**.

También encontramos aquí la selección del si se usará este módulo de forma síncrona (1) o asíncrona (0) con **UCSYNC**, y en el caso de ser asíncrona, podemos elegir entre 4 modos diferentes mediante**UCMODEx** en los que encontramos en orden de 0 a 3: modo UART, modo multiprocesador de linea activa, modo multiprocesador de direcciones de bit o modo UART con detección automática de la tasa de baudios.

5.5.2 UCAxCTL1

Registro de lectura duplicado igual que el anterior, que en este caso tendrá configuraciones como la selección del reloj fuente de este módulo con **UCSSELx** (0 - UCLK, 1 - ACLK , 2 y 3 -SMCLK), o el reseteo mediante software de este módulo con **UCSWRST**.

Otras configuraciones útiles se alojan en este registro como la habilitación de interrupciones con la llegada de caracteres erróneos mediante **UCRXEIE**.

5.5.3 UCAxBR0 y UCAxBR1

Ambos registros en conjunto forman el preescalado para el generador de la tasa de baudios, para poder conseguir una tasa diferente a la frecuencia marcada por el reloj utilizado. El factor de preescalado será de la forma (UCAxBR0 + UCAxBR1 × 256).

5.5.4 UCAxMCTL

En este registro se encuentran las opciones para configurar ambas etapas de modulación del módulo USCI_Ax bajo **UCBRFx** y **UCBRSx**.

5.5.5 UCAxRXBUF y UCAxRXBUF

En el primer registro se encuentra el último valor de la trama leído por el puerto de recepción tras pasar por el registro de desplazamiento propio de la comunicación asíncrona. En el segundo es el registro usado para enviar un carácter por el puerto de trasmisión.

5.5.6 IFG2

Este registro se utilizará para comprobar el estado de la trasmisión, ya que en el encontramos dos bits, **UCA0TXFG** y **UCA0RXFG**, los cuales nos indicarán con un 1 si hay alguna interrupción pendiente de saltar, o lo que es lo mismo si están siendo utilizados.

5.5.7 IE2

Gracias a este registro se podrá habilitar o deshabilitar las interrupciones asociadas a los puertos TX y RX mediante las máscaras **UCA0TXIE** y **UCA0RXIE**

5.6 Códigos

Una vez que se ha estudiado y comprendido la forma de configurar los diferentes módulos del MSP430, se pasará a mostrar los diferentes fragmentos de código utilizados a lo largo de las diferentes pruebas y del programa principal del control. Se irán explicando, conforme vaya surgiendo la necesidad, las diferentes configuraciones utilizadas, que se repetirán de un código al siguiente. Se incluirá en todos los códigos la cabecera propia del MSP430 en la cual se encontrarán las diferentes máscaras necesarias para la configuración de todos los módulos. Esto se hará añadiendo al comienzo de todos los archivos de código C:

```
1 #include <msp430.h>
```

5.6.1 Monitorización de temperaturas con el cerramiento

Para las pruebas realizadas en el apartado 4.1 se utilizaron tanto el conversor AD como la UART para transmitir los datos del MSP430 al ordenador. Para el uso de la UART se ha utilizado la librería “uart_STDIO” creada por M. Perales para la realización de otros proyectos que nos facilitó durante la asignatura de Sistemas Electrónicos en el tercer año del grado. Esta librería contiene varios métodos para facilitar tanto la iniciación

de la USCIA_0 como para la trasmisión de datos por el puerto serie en distintos formatos. En primer lugar se configurará la UART con el método *UARTinit()*

```
void UARTinit(void){

    P1SEL2|= BIT1 | BIT2;           // P1.1, P1.2 para la UART
    P1SEL|= BIT1 | BIT2;            // P1.1 --> RX y P1.2 --> TX

    UCA0CTL1 |= UCSWRST;            // Reset uart para configurar
    UCA0CTL1 = UCSSEL_2 | UCSWRST; // Conf. como asincrono
    UCA0MCTL = UCBRF_0 | UCBRS_1;  // Modulacion
    UCA0BR0 = 130;                  // 16MHz/(6*256 + 130)=9603,8415..
    UCA0BR1 = 6;
    UCA0CTL1 &= ~UCSWRST;           // Quitar Reset
    IFG2 &= ~(UCA0RXIFG);           // Borrar flag
    IE2 &=~ UCA0RXIE;               // Deshabilitar las ints .

}
```

Con esta función se conseguirá de una sola llamada configurar el modulo USCIA_0 en modo UART, con una tasa de baudios de 9600, con un bit de stop y sin bit de paridad. Es interesante destacar que los registros de selección de tasa de baudios están configurados expresamente para el uso de un reloj de 16MHz, de forma que si se quiere hacer uso de esta librería hay que tener en cuenta la modificación de estos valores si se varía la frecuencia de reloj.

Por otra parte, entre todas las funciones disponibles en la librería, se usará la función *UARTprintCR(frase)*, la cual a su vez hará uso de *UARTprintc(c)*.

```
void UARTprintc(char c){

  while (!( IFG2 & UCA0TXIFG));
  UCA0TXBUF = c;

}

void UARTprintCR(const char *frase){

  while(*frase )UARTprintc(*frase++);
  UARTprintc(10);
  UARTprintc(13);

}
```

La primera de las funciones lo que hace es esperar a que la línea TX este libre mediante el bucle while, y una vez que lo esté copia el valor del carácter de entrada en el registro UCA0TXBUF para que comience su trasmisión.

En la segunda función se recibe una cadena de caracteres mediante un puntero al comienzo de esta, y mientras que el puntero no desborde, o lo que es lo mismo, que sigan quedando caracteres que trasmitir, hace llamadas a la primera función con cada carácter, incrementando el puntero en cada llamada. Una vez que todos los caracteres han sido enviados, esta función envía dos caracteres de control del cursor. Estos caracteres correspondientes a los códigos ASCII 10 y 13 son los de salto de línea y retorno de carro respectivamente.

En siguiente lugar, para la iniciación del resto de elementos necesarios se ha creado la librería “init_config” con el objetivo de reducir el código del programa principal quedando así más legible. En esta librería se iniciarán, entre otros, los módulos del reloj y del conversor AD. Con la función *conf_reloj(VEL)* se inicializa el reloj a la frecuencia marcada por VEL, siendo posible seleccionar 1MHz, 8MHz, 12MHz y 16MHz.

```
void  conf_reloj (char  VEL){

  BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
  switch(VEL){

  case  1:
    if  (CALBC1_1MHZ != 0xFF) {
```

```
        DCOCTL = 0x00;
        BCSCTL1 = CALBC1_1MHZ;     /* Set DCO to 1MHz */
        DCOCTL = CALDCO_1MHZ;
      }
      break;
    case 8:

      if (CALBC1_8MHZ != 0xFF) {
        __delay_cycles(100000);
        DCOCTL = 0x00;
        BCSCTL1 = CALBC1_8MHZ;     /* Set DCO to 8MHz */
        DCOCTL = CALDCO_8MHZ;
      }
      break;
    case 12:
      if (CALBC1_12MHZ != 0xFF) {
        __delay_cycles(100000);
        DCOCTL = 0x00;
        BCSCTL1 = CALBC1_12MHZ;    /* Set DCO to 12MHz */
        DCOCTL = CALDCO_12MHZ;
      }
      break;
    case 16:
      if (CALBC1_16MHZ != 0xFF) {
        __delay_cycles(100000);
        DCOCTL = 0x00;
        BCSCTL1 = CALBC1_16MHZ;    /* Set DCO to 16MHz */
        DCOCTL = CALDCO_16MHZ;
      }
      break;
    default :
      if (CALBC1_1MHZ != 0xFF) {
        DCOCTL = 0x00;
        BCSCTL1 = CALBC1_1MHZ;     /* Set DCO to 1MHz */
        DCOCTL = CALDCO_1MHZ;
      }
      break;

    }
    BCSCTL1 |= XT2OFF | DIVA_0;
    BCSCTL3 = XT2S_0 | LFXT1S_2 | XCAP_1;
}
```

Como se puede observar esta función toma por defecto el valor de 1MHz aunque en nuestro caso se utilizará la máxima velocidad disponible de 16MHz.

La próxima función que se va implementa en esta librería será la de configuración del conversor analógico-digital con *init_ADC()*.

```
int BufferADC[4];

void init_ADC(void){

  P1DIR &= ~BIT3 | ~BIT4 | ~BIT5 | ~BIT6 ;
  ADC10CTL0 &= ~ENC;            //  deshabilita ADC
  /* varios canales ,      reloj ADC10OSC, sin preescalado, canal 6(primero a leer )*/
  ADC10CTL1 = CONSEQ_1 | ADC10SSEL_3 | ADC10DIV_0 | SHS_0 | INCH_6;
  /* Encender ADC, Conv. Multiple ,      S/H:x64,      V+=Vcc V-=gnd*/
  ADC10CTL0 = ADC10ON | MSC | ADC10SHT_3 | SREF_0;
  ADC10AE0 = 0x78;                  //  0x78=01111000: activar entradas 6, 5, 4, 3
  ADC10DTC0 = ADC10CT;              //  transferencia Continua: se resetea solo cuando acaba
  ADC10DTC1 = 7;                    //  Numero de transferencias a hacer
  ADC10SA = (short)&BufferADC[0]; // Direccion de comienzo de la transferencia
  ADC10CTL0 |= ENC;                 //  habilita ADC

}
```

Previo a la descripción de esta función habrá que declarar un array de enteros donde se quieran guardar las conversiones realizadas, ya que el conversor se inicializará de modo que haga múltiples conversiones,

comenzando por el canal 6 hasta llegar al canal 3. Los valores de estas conversiones serán automáticamente volcados en el array BufferADC.

Por último, una vez que tenemos todas las configuraciones iniciales declaradas, se pasa a escribir el código del programa principal, en él habra que incluir las cabeceras correspondientes a las librerías mencionadas, así como una adicional utilizada para crear cadenas de caracteres con formato. También será necesario declarar el array de enteros para la lectura de los sensores, y declarar las llamadas a las funciones auxiliares que se van a utilizar.

```
#include <msp430.h>
#include "uart_STDIO.h"
#include " init_config .h"
#include <stdio .h>

void run_ADC(int * temp, int length);
int BufferADC[4];

int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Parar el temporizador watchdog

    int i;                  // Contador auxiliar

    conf_reloj (16);        // Configurar reloj DC0 a 16MHz
    init_ADC();             // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
    UARTinit();             // Iniciar el puerto serie 9600 8B N 1bs
                            // P1.1 -> RX
                            // P1.2 -> TX
    int tmp[4];
    char frase [5];

    while(1){
        run_ADC(tmp,4);

        for(i=0;i<4;i++){
            sprintf ( frase ,"T\%d: \%d",(i+1), tmp[i]);
            UARTprintCR(frase);
        }

        UARTprintCR("");
        __delay_cycles(8000000);                    // Esperar 0.5s 8000000/16000000=0.5
    }

  return 0;
}
```

En esta función principal en primer lugar se parará el watchdog o timer de perro guardián, el cual viene configurado por defecto para que se reinicie el sistema si ha sucedido algún error y no se para. También se hará la llamada a las funciones de iniciación y se declararán un array en el que se almacenarán los valores de temperatura una vez realizado el cálculo de la temperatura asociada a la lectura del conversor y una cadena de caracteres con la cual iremos pasando por la UART los distintos valores de temperatura. Una vez hecho esto se llamará a la función *run_ADC(temp, length)* con la cual lanzaremos la conversión y una vez realizada transformaremos las lecturas del conversor a temperaturas. Tras esto se envían por la UART los valores obtenidos y esperaremos medio segundo. La espera de medio segundo se hará para evitar tener excesivas lecturas, ya que el sistema que se va a medir tiene una evolución relativamente lenta.

```
void run_ADC(int * temp, int length){

    int i;
    ADC10CTL0 |= ADC10SC;
    while(ADC10CTL1 & ADC10BUSY) /*EMPTY*/; // Esperar al fin de la conversion
    for(i=0; i<length;i++){
            temp[i] = ((BufferADC[i])*500)>>10;
    }
}
```

La función *run_ADC(temp, length)* como se puede observar coge el array declarado en main y lo rellena con los valores e la conversión tras la operación $LecturaADC * 500/1024$, para la cual se aprovecha que la división se hace por un número potencia directa de 2 y en vez de realizar la división, que es más costosa para la CPU, se realiza un desplazamiento de 10 bits a la derecha mediante el operador $>>$, lo cual es equivalente.

Es interesante observar que aunque la conversión contiene 10 bits de precisión, se utilizan números enteros, y esto es debido a que la precisión del sensor es de 1ºC, con lo cual, la medida de los decimales obtenida es despreciable.

5.6.2 Pruebas con la celda peltier sola

Para las pruebas realizadas con la celda peltier con y sin las aletas, se hará uso de todos los códigos anteriores a excepción del código principal, el cual variará. Además, para este caso también se hará uso del temporizador para generar señales PWM y se utilizarán un pin del puerto 2 como salida digital. Estas configuraciones se añadirán al código principal.

```
int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    int i, j;
    P2DIR |= BIT1 | BIT3; // P2.1 y P2.3 como salida
    P2SEL |= BIT1;          // Asociado al Timer_A1

    P2OUT |= BIT3;

    conf_reloj(16);         // Configurar reloj DC0 a 16MHz
    init_ADC();             // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
    UARTinit();             // Iniciar el puerto serie 9600 8B N 1bs
                            // P1.1 -> RX
                            // P1.2 -> TX
    int duty1 = 0;

    TA1CCR0 = 1000;   // Cargamos el periodo PWM
    TA1CCR1 = duty1;  // PWM duty cycle
    TA1CCTL1 = OUTMOD_7; //Modo7 reset/set
    TA1CTL = TASSEL_2 + MC_1; // Timer SMCLK Modo UP

    int tmp[2];
    char frase[5];

    while(1){

        for(j=0;j<100;j++){
            run_ADC(tmp,2);

            for(i=0;i<2;i++){
                sprintf(frase,"T\%d: \%d",(i+1), tmp[i]);
                UARTprintCR(frase);
            }

            UARTprintCR("");
            __delay_cycles(8000000);      // Esperar 0.5s 8000000/16000000=0.5
        }

        if(duty1<1000){
            duty1 += 200;           // Incrementar duty cycle en un 20\%
            TA1CCR1 = duty1;        // Actualizar el duty cycle
        }
    }

  return 0;
}
```

En este caso se asignarán los pines P2.1 y P2.3 como salidas, siendo el primero usado para generar la señal PWM y el segundo para activar el puente en H. Para esta prueba se van a realizar incrementos del duty cycle de 20 % en 20 % desde 0 hasta 100 % cada 50s, de forma que se puedan observar los cambios de

temperatura que se crean en ambas caras de la celda peltier. Además se irán realizando cada 0,5s mediciones de los sensores colocados en ambas cara de la celda peltier.

5.6.3 Control de la temperatura de los motores

Para el control de temperatura se volverá a hacer uso de todos los elementos vistos previamente, incluyendo en este caso el pin P2.0 como salida digital y el pin P2.4 como salida PWM para poder configurar ambas celdas peltier.

El código principal del control de la temperatura será el siguiente:

```
int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    int i;

    P2DIR |= BIT0 | BIT2 | BIT3 | BIT4 ; // P2.0, P2.1, P2.3 y P2.4 como salida
    P2SEL |= BIT2 | BIT4; // Asociado al Timer_A1

    P2OUT= BIT0 | BIT3; //Poner P2.0 y P2.3 a HIGH

    conf_reloj (16);          // Configurar reloj DC0 a 16MHz
    init_ADC();               // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
    UARTinit();               // Iniciar el puerto serie 9600 8B N 1bs
                              // P1.1 -> RX
                              // P1.2 -> TX


    int d1 = 0;
    int d2 = 0;

    TA1CCR0 = 1000; // Cargamos el periodo PWM
    TA1CCR1 = d1; // PWM 1 duty cycle
    TA1CCR2 = d2; // PWM 2 duty cycle
    TA1CCTL1 = OUTMOD_7; //Modo7 reset/set
    TA1CCTL2 = OUTMOD_7; //Modo7 reset/set
    TA1CTL = TASSEL_2 + MC_1; // Timer SMCLK Modo UP



    int tmp[4];
    char frase [5];

    while(1){

        run_ADC(tmp,4);

        for(i=0;i<4;i++){
            sprintf ( frase ,"T\%d: \%d",(i+1), tmp[i]);
            UARTprintCR(frase);
        }

        d1=get_Duty(tmp[0]);
        d2=get_Duty(tmp[1]);

  sprintf ( frase ,"D1: \%d", d1);
        UARTprintCR(frase);
         sprintf ( frase ,"D2: \%d", d2);
        UARTprintCR(frase);

        UARTprintCR("");

        TA1CCR1 = d1;
        TA1CCR2 = d2;
        __delay_cycles(8000000);                    // Esperar 0.5s 8000000/16000000=0.5
    }

  return 0;
}
```

En esta función a cada ciclo se comprobará la temperatura de los distintos elementos (motor del eje X, motor del eje Y, motor del extrusor y temperatura del cerramiento medido junto a la pantalla LCD), y se asignara un ancho de pulso a las dos señales PWM que controlan las celdas peltier de los dos primeros motores. Para obtener el duty cycle de las señales PWM se llama a la función *get_Duty(tmp)* en la que se recibe la temperatura y se devuelve el duty cycle con el que se debe actuar sobre dicha temperatura. Su código se puede observar a continuación:

```
1  int get_Duty(const int tmp){
2      int d=0;
3
4      if(tmp>=90){
5          d=1000;
6      }
7      else if(tmp>=80){
8          d=750;
9      }
10     else if(tmp>=70){
11         d=500;
12     }
13     else if(tmp>=60){
14         d=250;
15     }
16
17     return d;
18 }
```

6 Conclusiones

A lo largo de este trabajo se ha tratado de abordar los problemas con los que personalmente me he topado tras adquirir una de las impresoras 3D más baratas del mercado, la cual se corresponde con un modelo derivado del diseño original Prusa i3. Este modelo difiere en pequeños detalles con el resto máquinas descendientes de la Pursa i3, como el acabado del marco de la impresora o ciertas piezas correspondientes al ensamblaje de la impresora. Pero con todo y con esto, las dimensiones totales y la disposición de los diferentes motores son prácticamente idénticas en todas ellas. También en todas ellas son comunes los problemas tratados en el apartado 1.3.1, con este trabajo se consigue por tanto crear una solución económica para todas las personas que dispongan de un modelo descendiente de la Prusa i3 o semejantes, solución la cual en España tiene un coste total de 58,92€ contando con todos los materiales necesarios excepto con el coste del plástico utilizado. Además, se invita y recomienda a todo aquel que esté interesado en construir esta solución, que reutilice tantos elementos como pueda para la construcción del mismo. En mi caso personal, el presupuesto final ha sido algo menor a 46€ tras reciclar varios de los elementos utilizados.

Como se ha podido observar, el diseño final propuesto supone una importante mejora en las prestaciones de la impresora haciendo más sencillo su mantenimiento en cuanto a limpieza se refiere, quitando problemas como los de *warping* (ver figura 6.1) y permitiendo simultáneamente mantener la habitación de la impresora ventilada, reduciendo drásticamente la concentración de nanopartículas emitidas sin que esto comprometa en absoluto el resultado de la impresión. De esta forma esta solución no solo es beneficiosa para la impresora en sí, si no también para la salud de quien la utiliza. Además, aunque no era uno de los fines perseguidos, también se ha podido notar con el cerramiento una disminución del ruido que genera la máquina en funcionamiento, lo cual supone también un alivio para el resto de personas que conviven con la máquina.

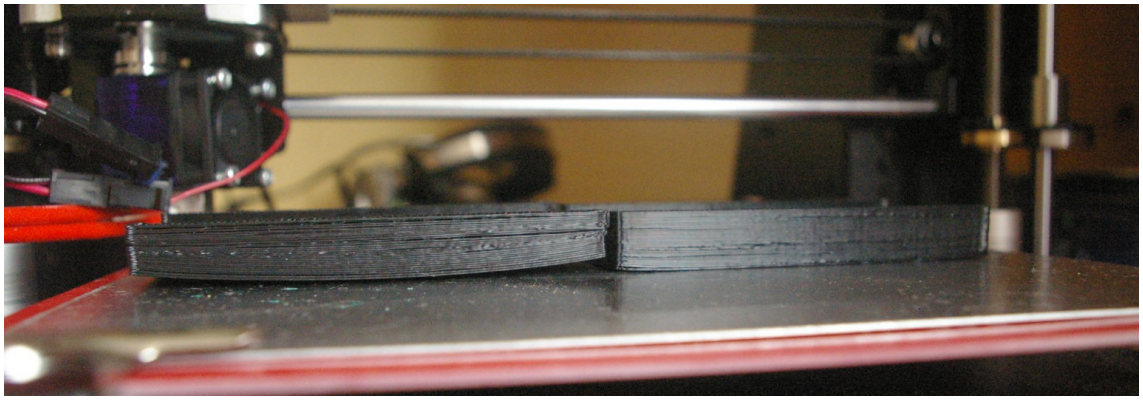


Figura 6.1 Resultado con adhesivo en la cama y sin cerramiento (izquierda), y sin adhesivo y con cerramiento (derecha).

Por otra parte, este diseño ha sido concienzudamente realizado para evitar que el propio cerramiento suponga un riesgo para la propia impresora. Como se puede ver en el próximo apartado, el riesgo detectado del sobrecalentamiento de los motores es evitado por completo.

6.1 Resultados finales con refrigeración

Tras haber detectado el problema del sobrecalentamiento de los motores, y haber montado el sistema de refrigeración, se ha procedido a realizar la misma prueba que en el apartado 4.1.3, pero en este caso se ha elegido una impresión de mayor volumen, para la cual se utilizará la configuración disponible en la tabla 6.1.

Tabla 6.1 Configuración para la impresión de prueba con ABS.

Parámetro	Unidad	Valor
Temperatura de la cama	ºC	110
Temperatura de la boquilla	ºC	245
Velocidad de impresión	mm/s	80
Velocidad de perímetro exterior	mm/s	70
Velocidad de relleno	mm/s	120
Densidad de relleno	%	100

Como se puede observar en ésta, se aumentarán todas las velocidades de impresión con respecto a las velocidades usadas para la prueba realizada sin refrigeración. Esta velocidad será ligeramente superior a la máxima por defecto. Por otra parte se aumenta la densidad de relleno al 100 %. Con esta configuración se trata de probar la impresora en una situación límite. Esta impresión dura un total de 5 horas, durante las cuales se registran de nuevo las temperaturas del motor del eje X, del eje Y, del extrusor, y el aire del cerramiento midiendo ésta ultima junto a la pantalla LCD de la impresora. En la gráfica de la figura 6.2 se pueden observar las temperaturas alcanzada, y en la siguiente figura, las señales PWM enviadas al módulo L298N.

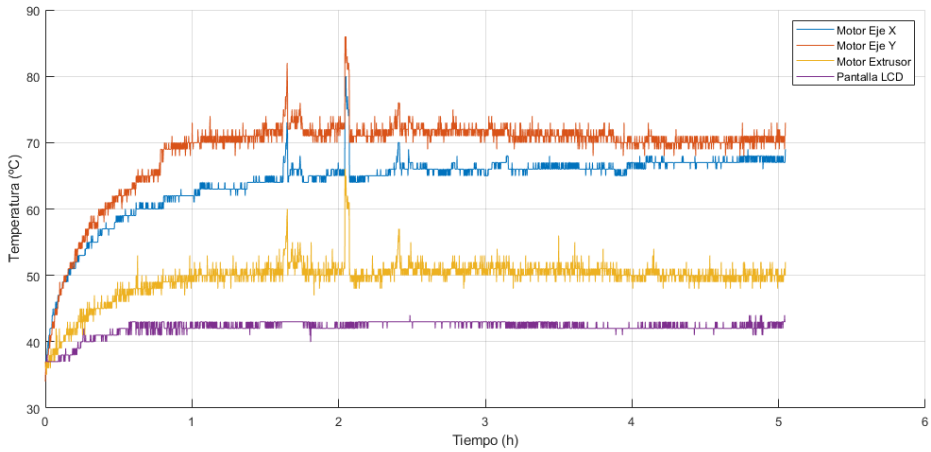


Figura 6.2 Gráfica de temperaturas con refrigeración.

En la primera gráfica de la evolución de las temperaturas, salta a la vista que el riesgo que supone mantener los motores sobre los 80ºC desaparece, dado que con la ayuda de la trasmisión de calor de las celdas peltier, los motores no sobrepasan en ningún momento los 73ºC de máxima. Además, a pesar de que la configuración de las velocidades utilizada, como se puede observar en la gráfica de los duty cycles de ambas señales PWM, ninguna de las celdas peltier trabaja a pleno rendimiento, trabajando prácticamente en todo momento al 25 % para el caso del eje X y al 50 % en el eje Y. Esto supone un resultado bastante positivo, ya que significa que se podrían llegar a utilizar valores de velocidad mayores en futuras impresiones, o que incluso se podría

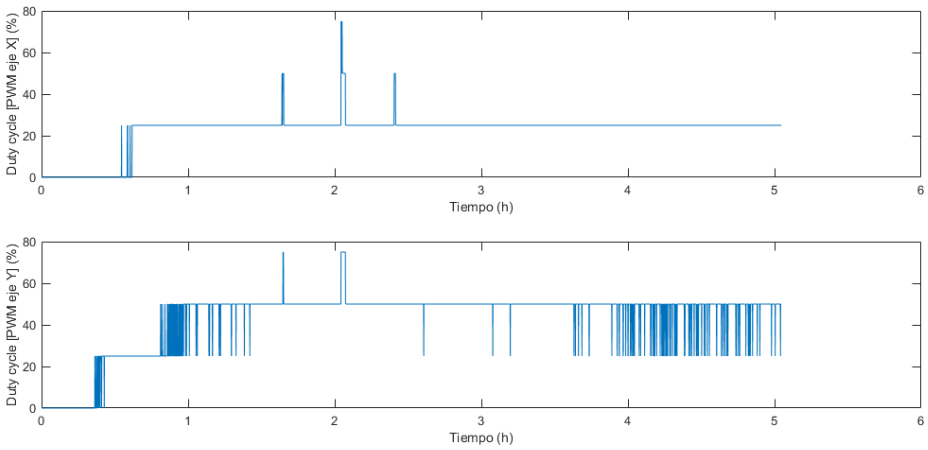


Figura 6.3 Gráfica de temperaturas con refrigeración.

trabajar con esta impresora en entornos de mayor temperatura ambiente (en el caso de este experimento, la temperatura de la habitación era de aproximadamente 29ºC).

La gran diferencia con respecto a la prueba realizada sin refrigeración reflejada en el apartado 4.1.3 es la brusca disminución de la pendiente de la temperatura de los motores del eje X e Y, llegando a tener pendiente ligeramente negativa en el motor del eje Y transcurridas las 3 horas de impresión. En el eje X la temperatura se puede observar creciente, siendo esta pendiente muy pequeña. Además, si llegara a superar los 70ºC, se podría observar una total desaparición de dicha pendiente al aumentar el rendimiento de la celda peltier asociada.

También se puede observar con respecto al primer experimento realizado, que al utilizarse una densidad de relleno del 100 %, el motor del extrusor se mantiene mucho más tiempo en movimiento, lo que supone mayor temperatura. Esto no supone un problema, dado que la temperatura a la que se estabiliza no supera los 54ºC. Esta diferencia tan brusca con respecto a los otros motores es debida a la diferencia de velocidades y aceleraciones a las que operan los motores del eje X e Y con respecto al extrusor.

Por otro lado, la temperatura del aire del cerramiento se vuelve a estabilizar a la misma temperatura a la que lo hacía en el primer caso (43ºC). Esto no supone un problema para el resto de dispositivos electrónicos como lo son la controladora de la impresora o los drivers de los motores dado que estos elementos por el propio diseño de la impresora se encuentran refrigerados con ventiladores. Con estos junto con la temperatura que hace en el cerramiento, dichos componentes electrónicos se encuentran en condiciones seguras.

6.2 Futuras mejoras posibles

La versatilidad del MSP430 es indiscutible. Con la variedad de interfaces de comunicación que este dispone abre un gran abanico de posibilidades en cuanto a la integración del sistema de control de temperatura con otras tecnologías que envuelvan a la de impresión 3D. Durante el trascurso del trabajo o mientras he investigado en formas de mejorar o hacer más accesible la impresora, múltiples ideas se me vinieron a la cabeza, las cuales finalmente no he llevado al ámbito de este trabajo por falta de tiempo y por no hacer del TFG un trabajo de mayores dimensiones. Entre estas ideas, se han seleccionado algunas de ellas que pueden ser de interés para muchas personas y que supone una continuación de este trabajo.

6.2.1 Integración con OctoPrint

OctoPrint es una herramienta creada por Gina Häußge con la cual gracias a una Raspberry Pi (recomendable usar el modelo Pi 3 o superior) se puede conseguir una interfaz web con la que controlar y visualizar una impresora 3D. Ésta es una herramienta de código abierto, lo cual ha permitido que se creen multitud de plugins con los cuales añadir nuevas capacidades a dicho software.

Figura 6.4 Logo de OctoPrint.

Gracias a la comunicación serial ya implementada con el MSP430 y a las compatibilidades que ofrece esta herramienta, sería relativamente sencillo integrar el monitorizado de las diferentes temperaturas tratadas en el trabajo, así como las variables de control enviadas hacia el módulo L298N, permitiendo la visualización remota de los diferentes elementos del trabajo.

6.2.2 Integración con MQTT

La domótica es otro campo de la tecnología que hoy día se encuentra en pleno crecimiento, por ello podría ser interesante ampliar la capacidad de comunicación del sistema creado mediante el uso del protocolo MQTT. Éste es un protocolo de comunicación maquina-a-maquina muy ligero, el cual permite una comunicación segura y sencilla con múltiples dispositivos a la vez. Este protocolo orientado al internet de las cosas es frecuentemente utilizado para sistemas de domótica de bajo coste. Sería necesario añadir algún microcontrolador, como por ejemplo el ESP8266, que disponga de una interfaz wifi y alguna interfaz de comunicación I^2C, SPI o UART. Se ha mencionado el ESP8266 por incluir dichas características por un precio de alrededor de los 4€.

Se podría comunicar el ESP8266, con un cliente MQTT implementado, con el MSP430, de manera que se pueda incluir el sistema de control de temperatura en cualquiera de los múltiples proyectos existentes de domótica. Además, añadiendo algún servo junto al interruptor, sería posible también controlar el encendido y el apagado de la impresora.

Apéndice A
Planos

En este apéndice se incluirán los planos de las piezas diseñadas más importantes para el desarrollo del trabajo con sus cotas

A.1 Sobre los planos

En este apéndice se incluirán los planos referentes a los cortes a realizar en los paneles de poliestireno, referentes a las piezas impresas indispensables para la realización del cerramiento y por último, los planos de los diferentes circuitos montados durante el trascurso del trabajo.

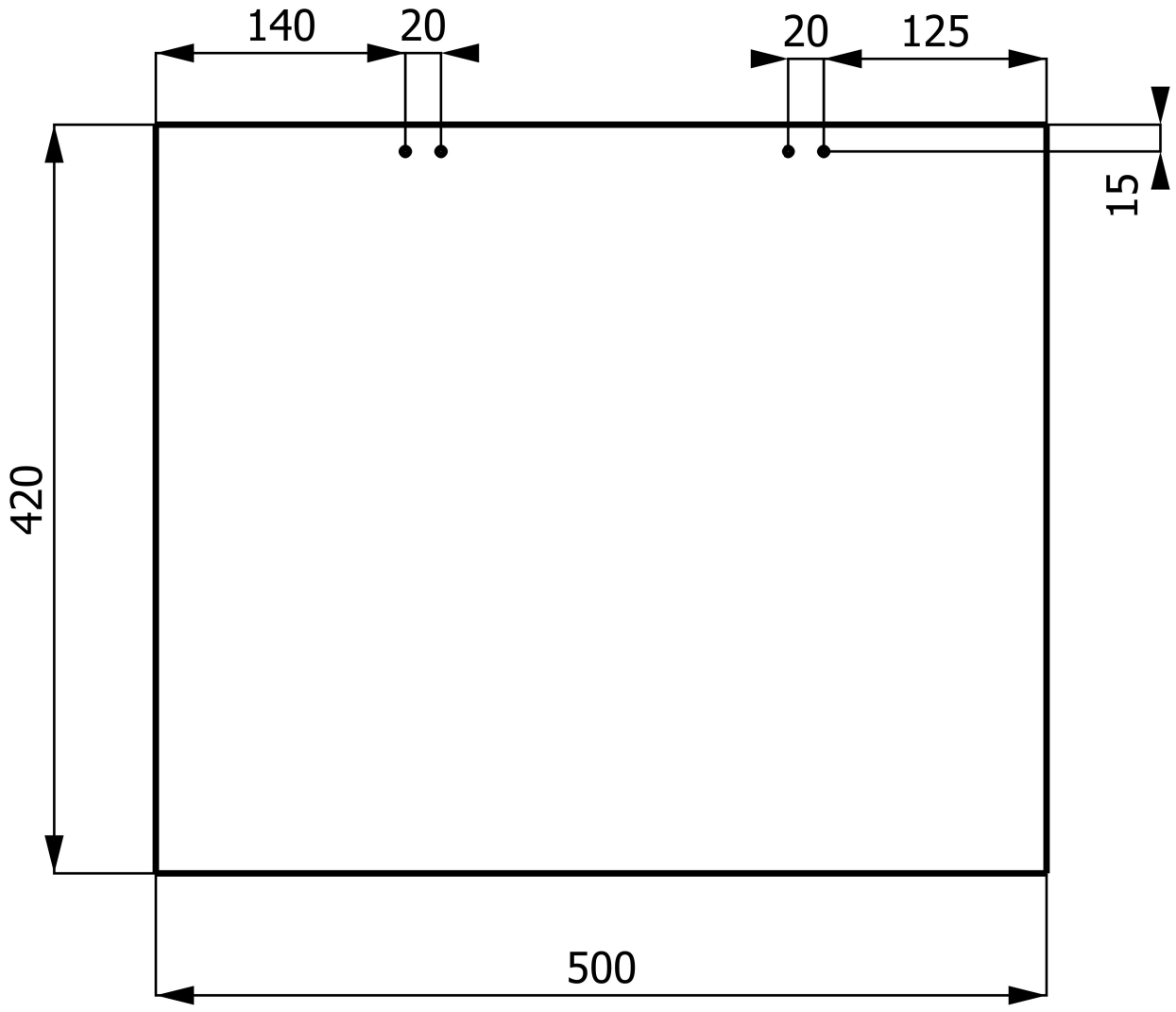
Tanto los planos de las piezas como de las planchas fueron realizados con FreeCad, haciendo uso de una plantilla que trae incorporado el programa, realizado según la norma ISO7200 para los márgenes y cajetín de los mismos. En ellos se encontrarán las vistas mas características de las piezas que se están acotando, así como vistas auxiliares en las piezas más complejas para ayudar a su comprensión.

Por otro lado, para la realización de los esquemas eléctricos se ha hecho uso del programa profiCAD. Con él se crearan tanto los símbolos de los diferentes elementos de los circuitos, como los propios esquemáticos. Nuevamente este software dispone de una versión gratuita gracias a la cual podremos realizar todo esto.

A.2 Planos de los paneles

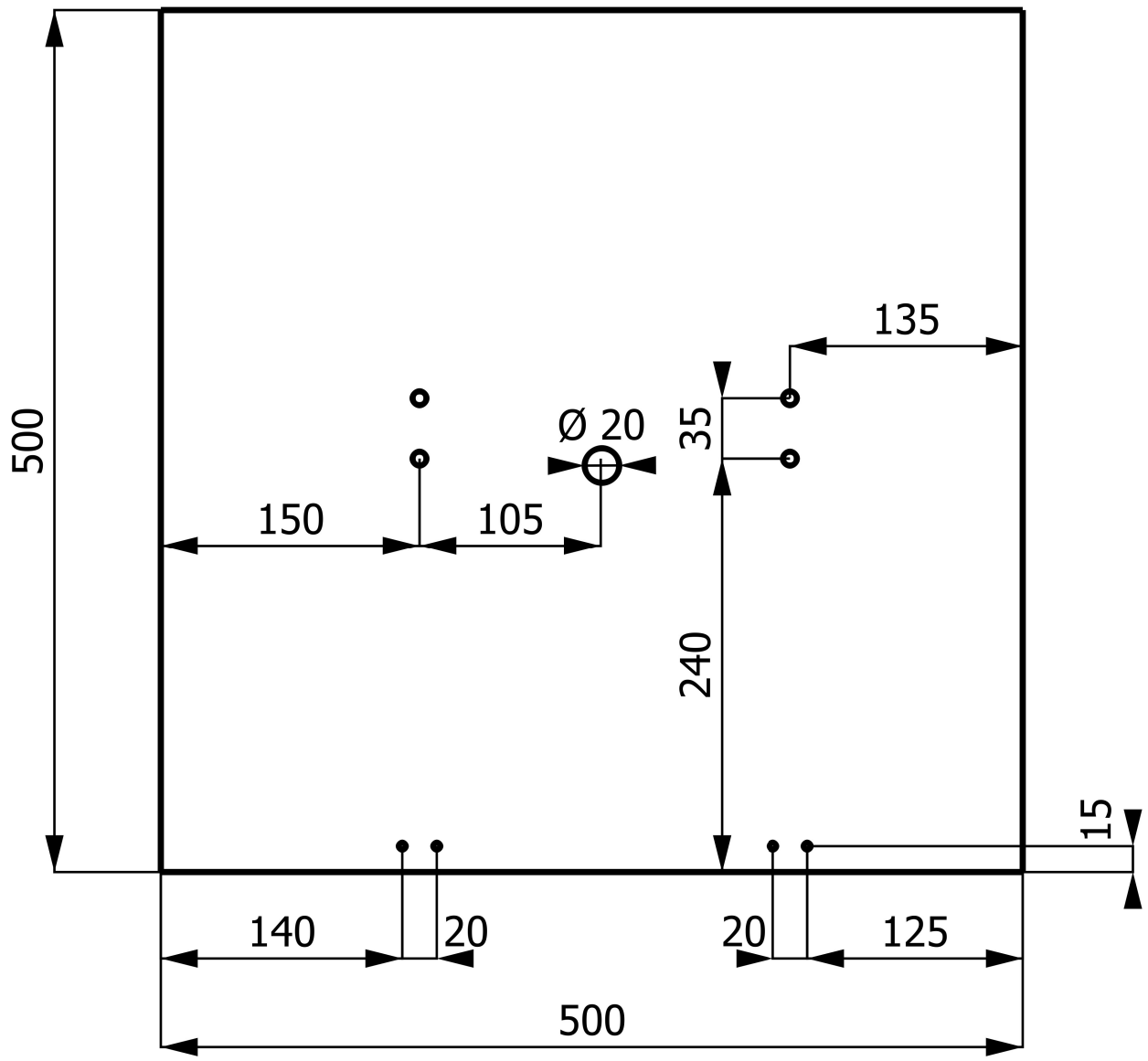
En los siguientes planos se muestran todas las medidas necesarias para realizar los cortes a las diferentes planchas de poliestireno, o el material elegido. Diferentes formas de conseguir estos cortes junto a más información sobre el montaje se abordan en el apartado 4.1.

Es importante mencionar que los diámetros de los orificios para los tornillos realizados a las planchas no quedan reflejados en los planos debido a su pequeño tamaño. Para ellos habrá que elegir un diámetro igual o ligeramente superior al de los tornillos utilizados para el ensamblaje, que en este caso, los orificios se han realizado de 3 mm de diámetro.



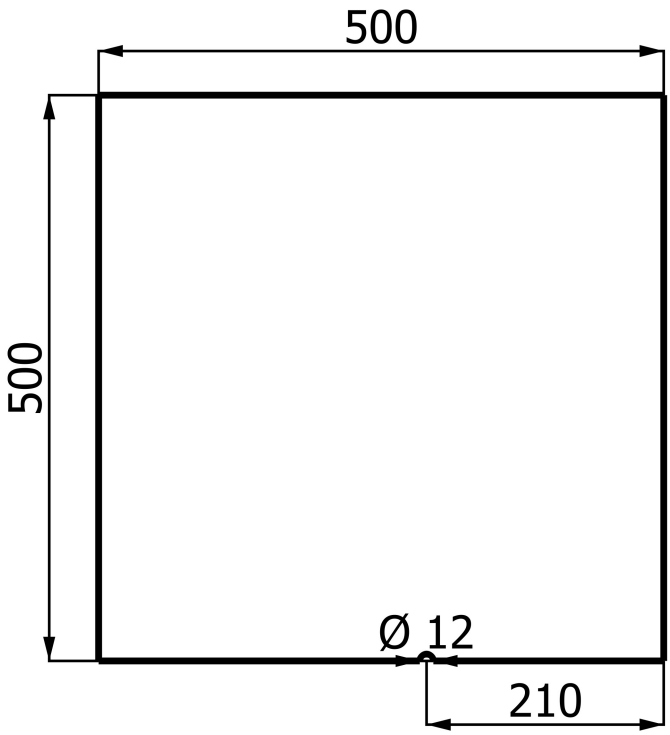
Autor:	Título:
Fidel González Leiva	Panel delantero

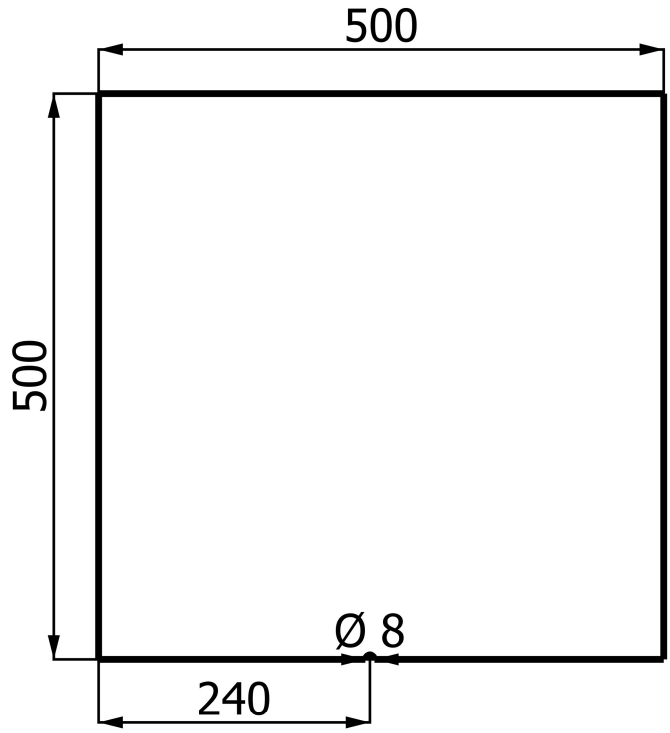
Información adicional:	Tamaño:	Unidades:	Escala:
Panel delantero del cerramiento con cotas	A4	mm	1:4
	Parte número: 1		
	Fecha: 12/08/2018		Revisión: REV A



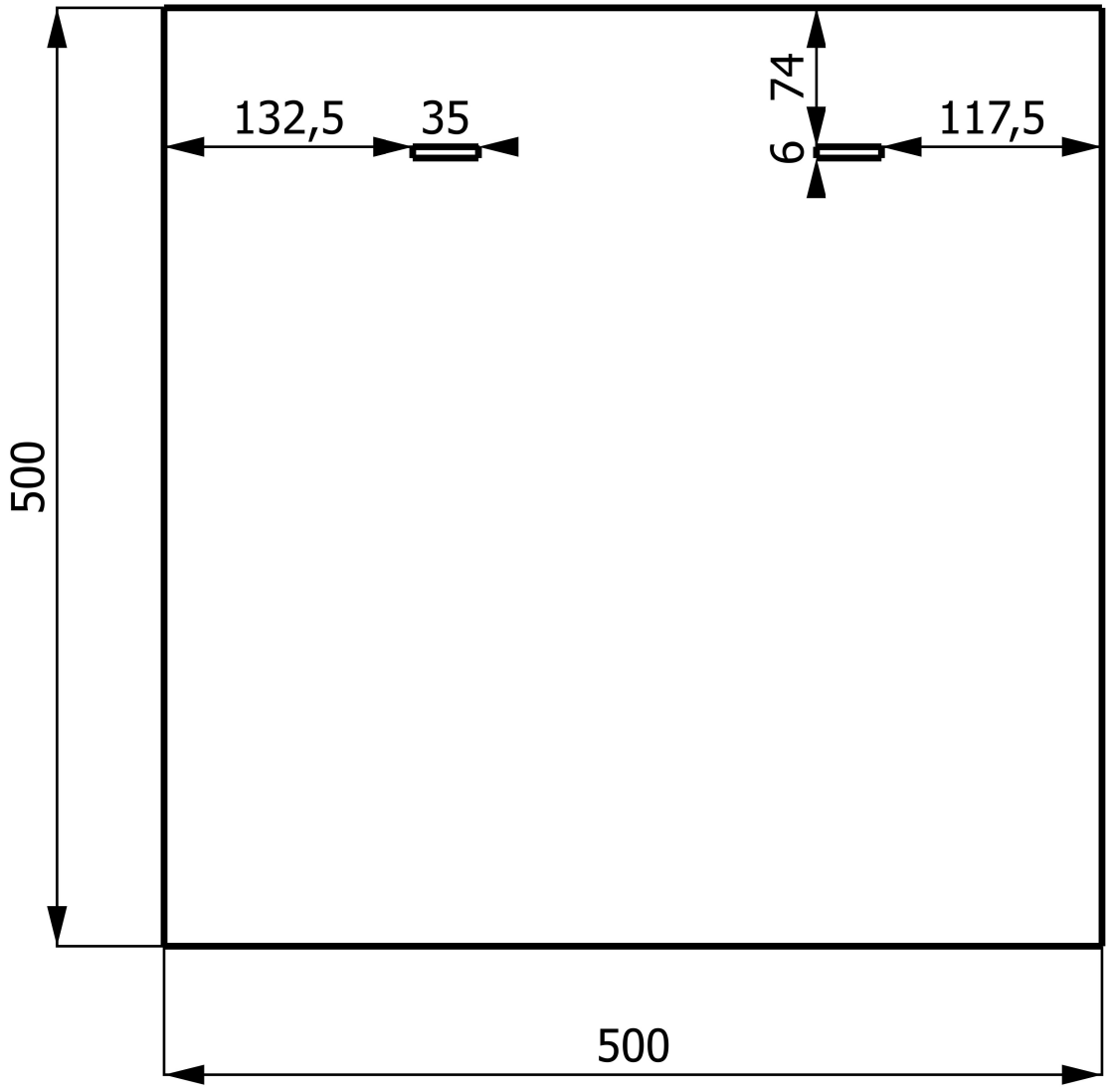
Autor:	Título:
Fidel González Leiva	Panel superior

Información adicional:	Tamaño:	Unidades:	Escala:
Panel superior del cerramiento con cotas	A4	mm	1:4
	Parte número: 2		
	Fecha: 12/08/2018		Revisión: REV A





Autor:	Título:			
Fidel González Leiva	Paneles laterales			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	3:20	
Panel derecho encima del panel izquierdo con cotas	Parte número: 3			
	Fecha:	Revisión:		
	12/08/2018	REV A		

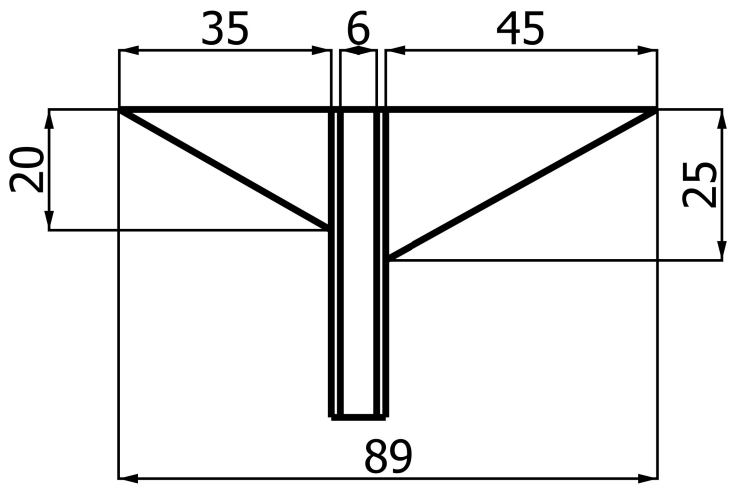


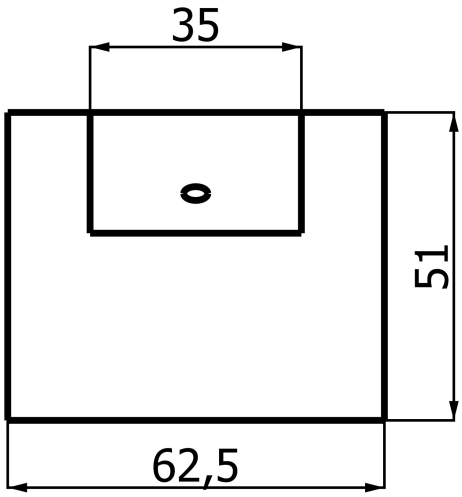
Autor:	Título:			
Fidel González Leiva	Panel trasero			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	1:4	
Panel trasero del cerramiento con cotas	Parte número: 4			
	Fecha: 12/08/2018		Revisión: REV A	

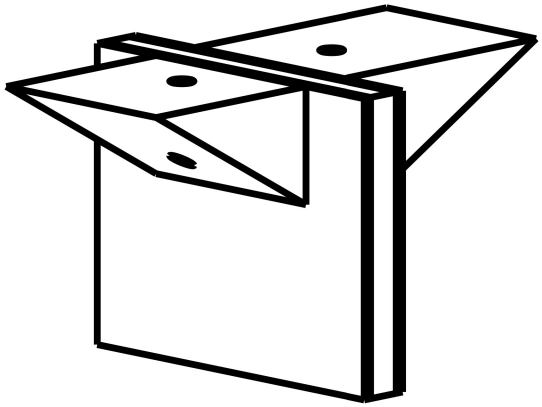
A.3 Planos de las piezas

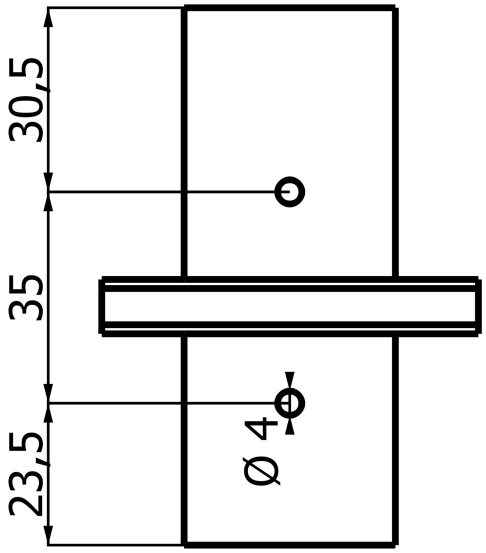
En este apartado se mostrarán las piezas diseñadas para ensamblar las planchas previamente vistas. En ellas quedarán reflejadas las dimensiones más relevantes de las piezas imprescindibles diseñadas, quedado excluidas las pinzas para los imanes por su simplicidad.

Más información sobre como ensamblar todas las piezas y detalles sobre el montaje se puede encontrar en el apartado 4.1.

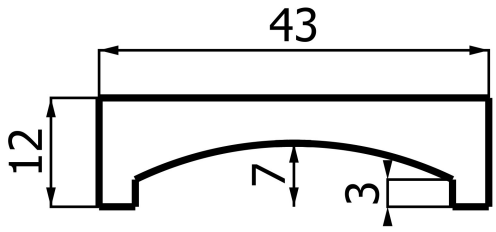


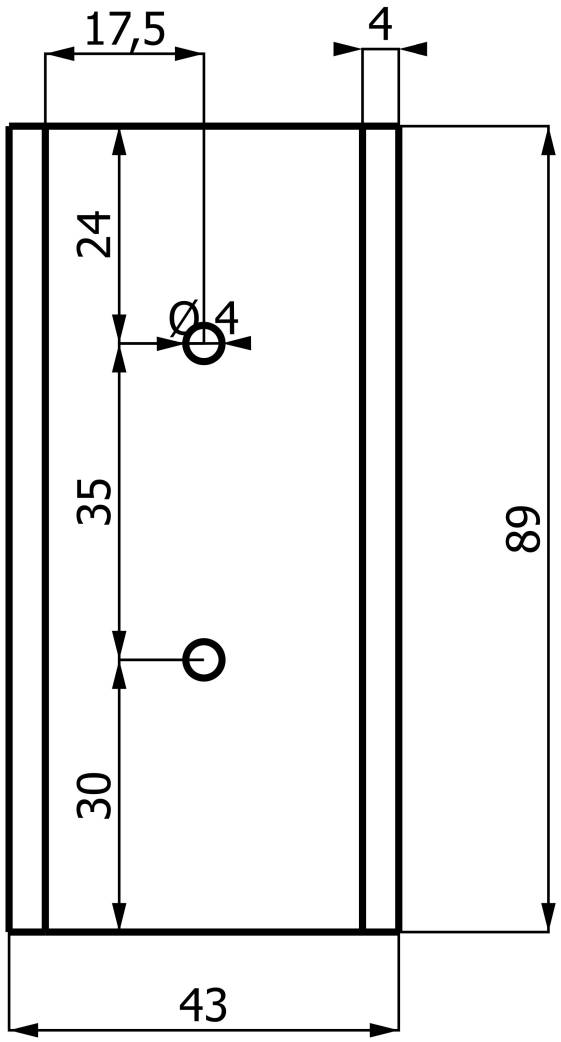




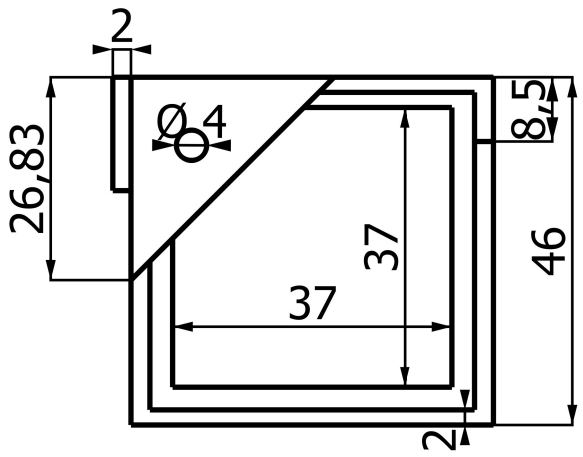


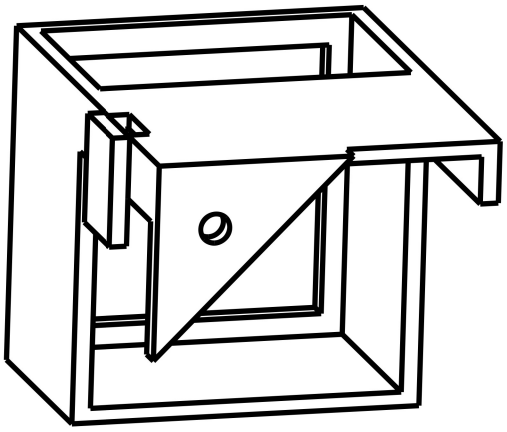
Autor:	Título:		
Fidel González Leiva	Pinza para el marco		
Información adicional:	Tamaño:	Unidades:	Escala:
	A4	mm	4:5
Pinza izquierda para el marco con cotas	Parte número: 1		
	Fecha: 18/08/2018		Revisión: REV A

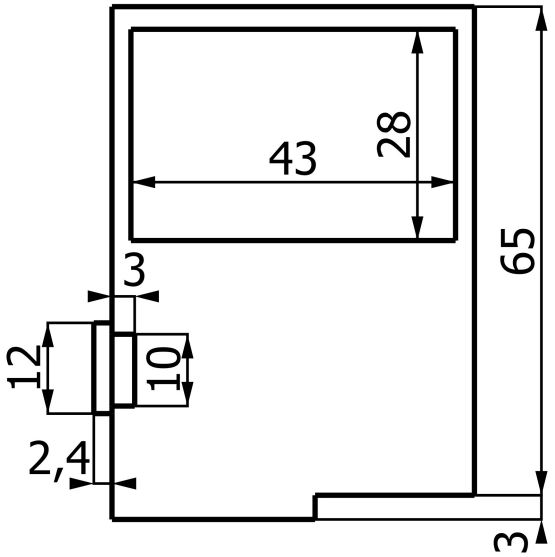


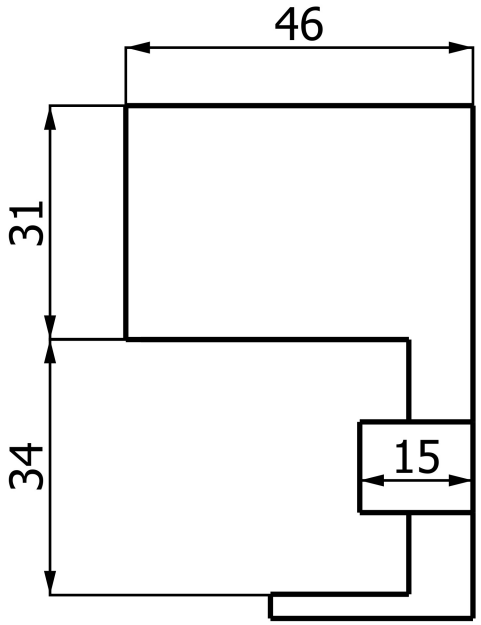


Autor:	Título:			
Fidel González Leiva	Abrazadera			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	6:5	
Abrazadera con cotas	Parte número: 2			
	Fecha: 18/08/2018		Revisión: REV A	

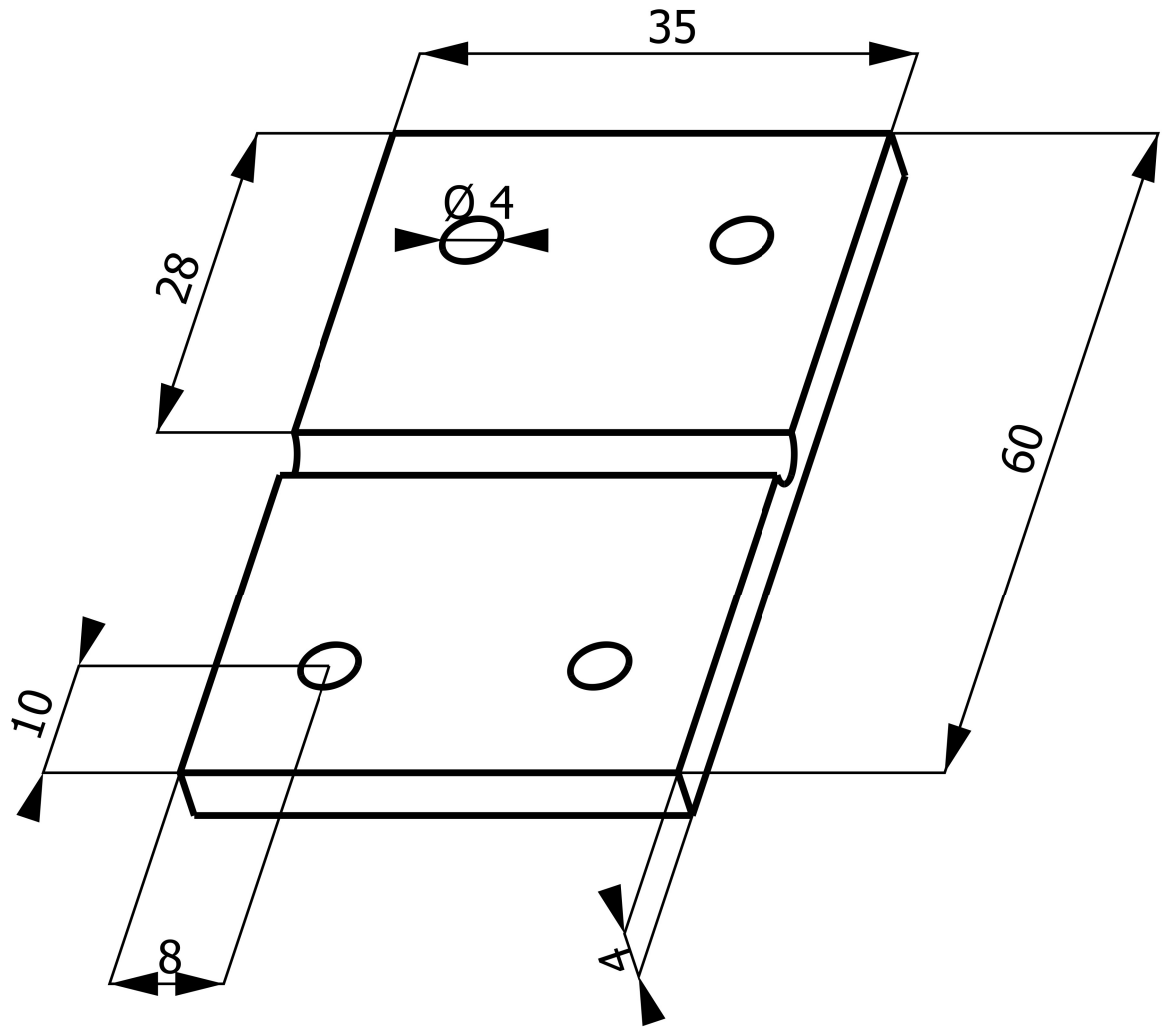




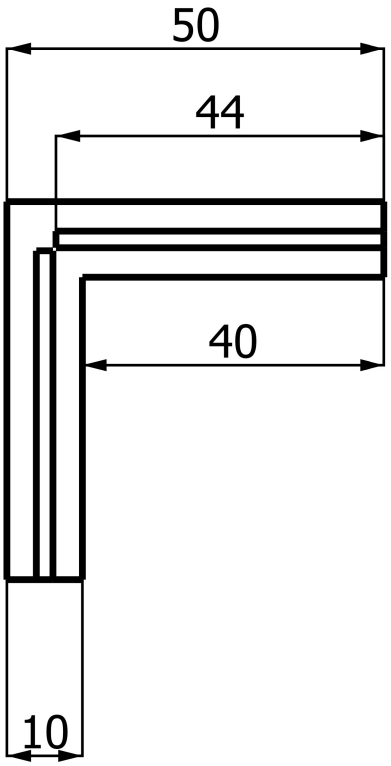


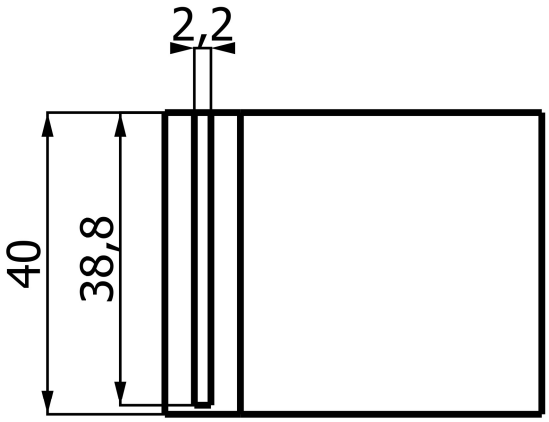


Autor:	Título:			
Fidel González Leiva	Soporte para motor			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	1:1	
Soporte para el motor del eje X con cotas	Parte número: 3			
	Fecha: 28/08/2018		Revisión: REV A	



Autor:	Título:			
Fidel González Leiva	Bisagra			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	2:1	
Bisagra flexible con cotas	Parte número: 4			
	Fecha: 28/08/2018		Revisión: REV A	

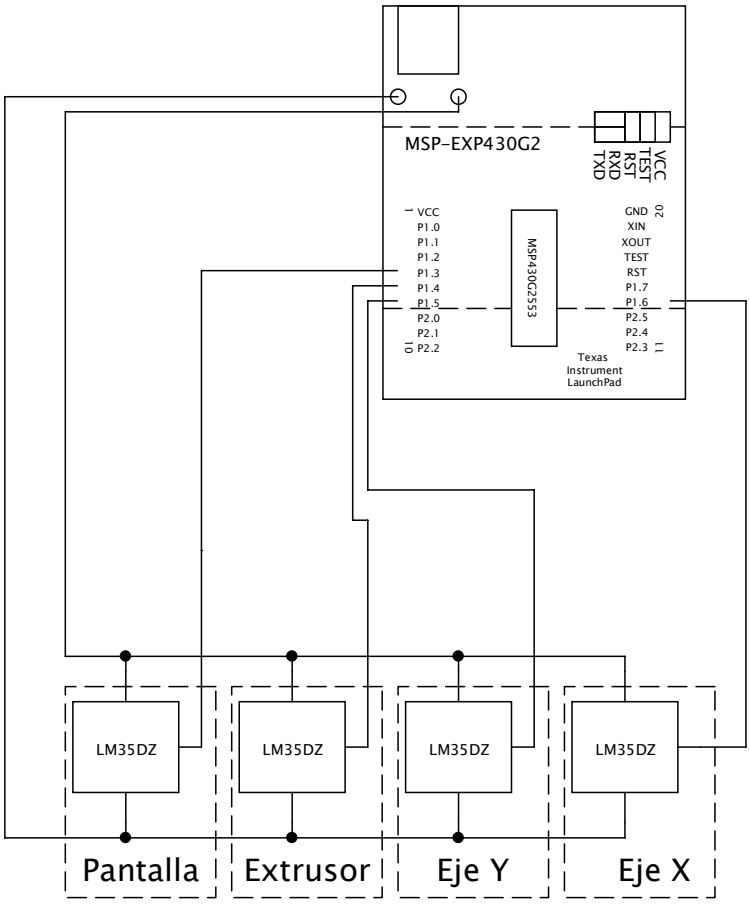




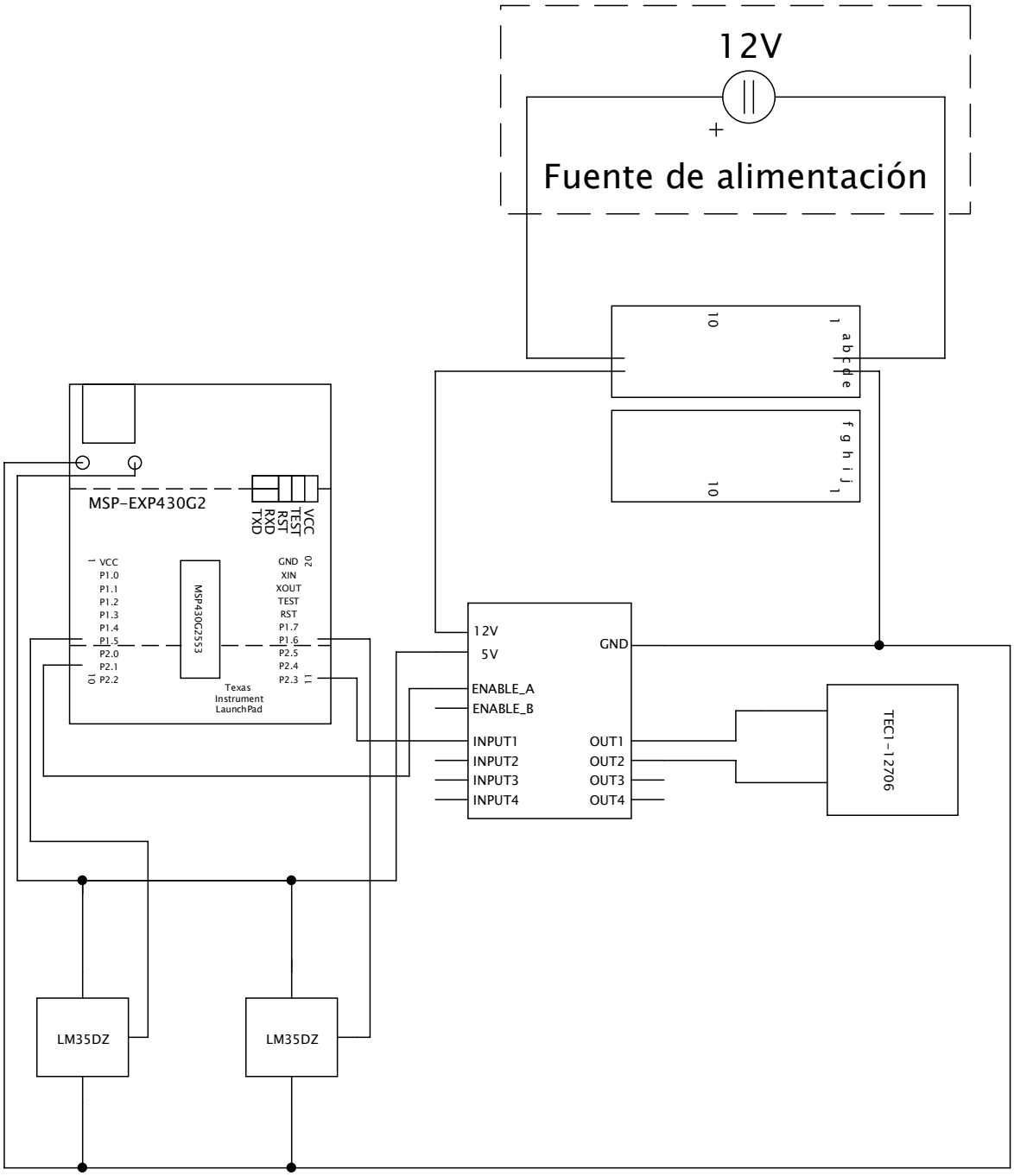
Autor:	Título:			
Fidel González Leiva	Escuadra trasera			
Información adicional:	Tamaño:	Unidades:	Escala:	
	A4	mm	1:1	
Escuadra trasera con cotas	Parte número: 5			
	Fecha: 28/08/2018		Revisión: REV A	

A.4 Planos de los circuitos

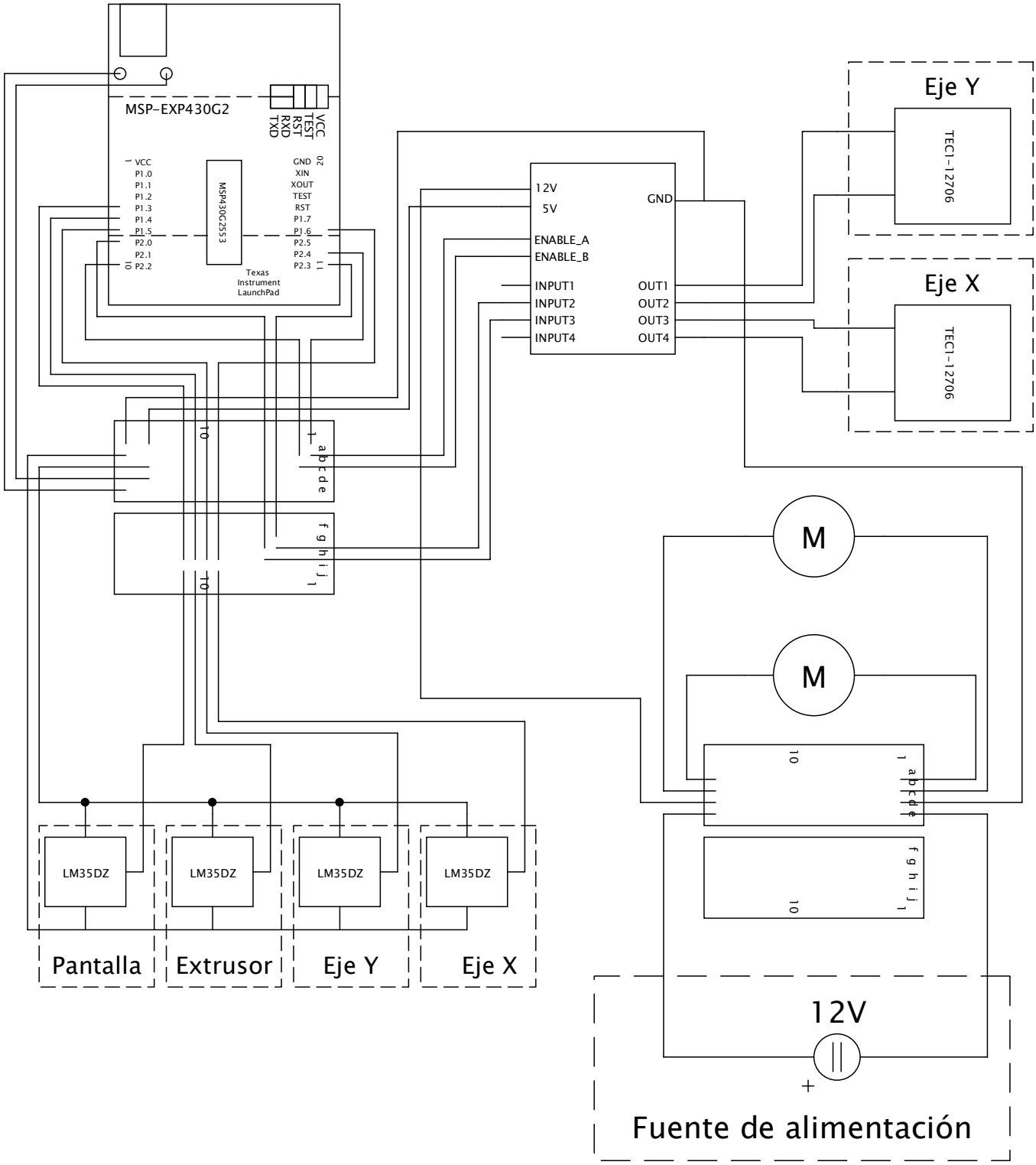
Estos planos se corresponden con las figuras 4.4, 4.7 y 4.13. Para comprender correctamente la disposición de los distintos elementos que se pueden encontrar en los esquemáticos es conveniente revisar el capitulo 4 y las figuras mencionadas.



profiCAD	Tipo de documento **Plano de conexionado**	Autor **Fidel González Leiva**	Rev. **A**
	Título, título suplementario **Esquema de conexionado** **Prueba de temperaturas**		
		Fecha de edición **7/09/2018**	Idioma **ES**
			Hoja **1**



profiCAD	Tipo de documento **Plano de conexionado**	Autor **Fidel González Leiva**	Rev. **A**
	Título, título suplementario **Esquema de conexionado** **Prueba con celdas peltier**	Fecha de edición **7/09/2018**	Idioma **ES** / Hoja **1**



profiCAD	Tipo de documento **Plano de conexionado**	Autor **Fidel González Leiva**	Rev. **A**
	Título, título suplementario **Esquema de conexionado** **Circuito de refrigeración**		
		Fecha de edición **7/09/2018**	Idioma **ES** / Hoja **1**

Apéndice B
Códigos completos

En este apéndice se adjuntaran todos los códigos de los distintos ficheros utilizados para hacerlos más accesibles, así como para facilitar la comprensión de estos

B.1 Librería uart_STDIO

B.1.1 Código C

```
/*
 * uart_STDIO.c
 *
 *  Created on: 15/12/2014
 *      Author: Manolo
 */
#include "uart_STDIO.h"

#include <msp430.h>


void UARTinit(void){
	P1SEL2|= BIT1 | BIT2;		// P1.1, P1.2 para la UART
	P1SEL|= BIT1 | BIT2;		// P1.1 --> RX y P1.2 --> TX

	UCA0CTL1 |= UCSWRST;		// Reset uart para configurar
	UCA0CTL1 = UCSSEL_2 | UCSWRST; // Conf. como asincrono
	UCA0MCTL = UCBRF_0 | UCBRS_1; // Modulacion
	UCA0BR0 = 130;			// 16MHz/(6*256 + 130)=9603,8415..
	UCA0BR1 = 6;
	UCA0CTL1 &= ~UCSWRST;		// Quitar Reset
	IFG2 &= ~(UCA0RXIFG);		// Borrar flag
	IE2 &=~ UCA0RXIE;		// Deshabilitar las ints .
}


void UARTprintc(char c){
  while (!( IFG2 & UCA0TXIFG)); //espera a Tx libre
		UCA0TXBUF = c;
}

void UARTprint(const char * frase ){
  while(* frase )UARTprintc(*frase++);
}

void UARTprintCR(const char *frase){
  while(* frase )UARTprintc(*frase++);
  UARTprintc(10);
  UARTprintc(13);

}

```

```
void UARTgets( char *BuffRx, int TMAX){
  int indice=0;
  char caracter ;
  do{
    while (!( IFG2&UCA0RXIFG));
      caracter =UCA0RXBUF;
    UARTprintc(caracter);
    if (( caracter !=10)&&(caracter!=13)){BuffRx[indice]= caracter ;
    indice++;}
  }while(( caracter !=13)&&(indice<TMAX));

  if ( indice ==TMAX)indice--;
  BuffRx[indice ]=0;
}


int UARTgetint(void){
    char caracter ;
    char Err=0;
    unsigned long num=0;
    unsigned int Num;
    do{
        while (!( IFG2&UCA0RXIFG));
        caracter =UCA0RXBUF;
        if ( caracter <'0' || caracter >'9' ) { if (( caracter !=10)&&(caracter!=13)){Err=1;}}
          else {
          num*=10;
          num+=(caracter-'0');
          if (num>0xffff) {num-=caracter-'0'; num/=10; Err=2;}}
        UARTprintc(caracter);
      }while(( caracter !=13)&&(Err==0));
    if (Err==1) { Num=0xffff;}
    else {Num=(int)num;}


  return (Num);
}


void UARTnum(int Num){
  char nch[]="0000";
    nch[0]+=Num/1000;
    Num=Num-1000*(Num/1000);
    nch[1]+=Num/100;
    Num=Num-100*(Num/100);
    nch[2]+=Num/10;
      Num=Num-10*(Num/10);
      nch[3]+=Num;
      UARTprint(nch);
}
```

B.1.2 Cabeceras

```
/*
 * uart_STDIO.h
 *
 *  Created on: 15/12/2014
 *      Author: Manolo
 */

void UARTinit(void);
void UARTprintc(char c);
void UARTprint(const char * frase);
void UARTprintCR(const char *frase);
void UARTgets( char *BuffRx, int TMAX);
int  UARTgetint(void);
void UARTnum(int Num);
```

B.2 Librería init_config

B.2.1 Código C

```
#include <msp430.h>
#include " init_config .h"


int BufferADC[4];

void init_ADC(void){
  P1DIR &= ~BIT3 | ~BIT4 | ~BIT5 | ~BIT6 ;
  ADC10CTL0 &= ~ENC;            //  deshabilita  ADC

  /* varios  canales ,      reloj  ADC10OSC, sin preescalado, canal  6(primero  a  leer ) */
  ADC10CTL1 = CONSEQ_1 | ADC10SSEL_3 | ADC10DIV_0 | SHS_0 | INCH_6;
  /* Encender ADC,  Conv. Multiple ,      S/H:x64,      V+=Vcc V-=gnd*/
  ADC10CTL0 = ADC10ON | MSC | ADC10SHT_3 | SREF_0;
  ADC10AE0 = 0x78;                  //  0x78=01111000: activar  entradas  6,  5,  4,  3
  ADC10DTC0 = ADC10CT;              //  transferencia  Continua:  se  resetea  solo  cuando acaba
  ADC10DTC1 = 7;                    //  Numero de  transferencias  a  hacer
  ADC10SA = (short)&BufferADC[0]; // Direccion de  comienzo de  la  transferencia
  ADC10CTL0 |= ENC;                 //  habilita  ADC
}

void  conf_reloj (char  VEL){
  BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
  switch(VEL){
  case  1:
    if  (CALBC1_1MHZ != 0xFF) {
      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_1MHZ;     /* Set DCO to 1MHz */
      DCOCTL = CALDCO_1MHZ;
    }
    break;
  case  8:

    if  (CALBC1_8MHZ != 0xFF) {
      __delay_cycles(100000);
      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_8MHZ;     /* Set DCO to 8MHz */
      DCOCTL = CALDCO_8MHZ;
    }
    break;
  case  12:
    if  (CALBC1_12MHZ != 0xFF) {
      __delay_cycles(100000);
      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_12MHZ;    /* Set DCO to 12MHz */
      DCOCTL = CALDCO_12MHZ;
    }
    break;
  case  16:
    if  (CALBC1_16MHZ != 0xFF) {
      __delay_cycles(100000);
      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_16MHZ;    /* Set DCO to 16MHz */
      DCOCTL = CALDCO_16MHZ;
    }
    break;
  default :
    if  (CALBC1_1MHZ != 0xFF) {
      DCOCTL = 0x00;
      BCSCTL1 = CALBC1_1MHZ;     /* Set DCO to 1MHz */
      DCOCTL = CALDCO_1MHZ;
    }
    break;

  }
```

```
66    BCSCTL1 |= XT2OFF | DIVA_0;
67    BCSCTL3 = XT2S_0 | LFXT1S_2 | XCAP_1;
68
69  }
```

B.2.2 Cabeceras

```
1  /*
2   * Cabezera de las configuraciones inicales de:
3   *     -> Reloj
4   *     -> ADC
5   */
6
7  void inicia_ADC(void);
8  void conf_reloj (char VEL);
```

B.3 Prueba de temperaturas sin control

```
1  #include <msp430.h>
2  #include "uart_STDIO.h"
3  #include " init_config .h"
4  #include <stdio .h>
5
6  void run_ADC(int * temp, int length);
7
8  int BufferADC[4];
9
10 int main(void) {
11     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
12
13     int i;                    // Contador auxiliar
14
15     conf_reloj (16);          // Configurar reloj DC0 a 16MHz
16     init_ADC();               // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
17     UARTinit();               // Iniciar el puerto serie 9600 8B N 1bs
18                               //  P1.1 -> RX
19                               //  P1.2 -> TX
20
21     int tmp[4];
22     char frase [5];
23
24     while(1){
25         run_ADC(tmp,4);
26
27         for(i=0;i<4;i++){
28             sprintf ( frase ,"T\%d: \%d",(i+1), tmp[i]);
29           UARTprintCR(frase);
30         }
31
32         UARTprintCR("");
33         __delay_cycles(8000000);                    // Esperar 0.5s 8000000/16000000=0.5
34     }
35
36   return 0;
37 }
38
39
40 void run_ADC(int * temp, int length){
41     int i;
42     ADC10CTL0 |= ADC10SC;
43     while(ADC10CTL1 & ADC10BUSY) /*EMPTY*/; // Esperar al fin de la conversion
44     for(i=0; i<length;i++){
45         temp[i] = ((BufferADC[i])*500)>>10;
46     }
47 }
```

B.4 Prueba de diferencia de temperatura de celda peltier

```
#include <msp430.h>
#include "uart_STDIO.h"
#include " init_config .h"
#include <stdio .h>

void run_ADC(int * temp, int length);

int BufferADC[4];

int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    int i, j;
    P2DIR |= BIT1 | BIT3; // P2.1 y P2.3 como salida
    P2SEL |= BIT1;        // Asociado al Timer_A1

    P2OUT |= BIT3;

    conf_reloj (16);      // Configurar reloj DC0 a 16MHz
    init_ADC();           // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
    UARTinit();           // Iniciar el puerto serie 9600 8B N 1bs
                          // P1.1 -> RX
                          // P1.2 -> TX
    int duty1 = 0;

    TA1CCR0 = 1000;  // Cargamos el periodo PWM
    TA1CCR1 = duty1;  // PWM duty cycle
    TA1CCTL1 = OUTMOD_7; //Modo7 reset/set
    TA1CTL = TASSEL_2 + MC_1; // Timer SMCLK Modo UP

    int tmp[2];
    char frase [5];

    while(1){

        for(j=0;j<100;j++){
            run_ADC(tmp,2);

            for(i=0;i<3;i++){
                sprintf ( frase ,"T\%d: \%d",(i+1), tmp[i]);
                UARTprintCR(frase);
            }

            UARTprintCR("");
            __delay_cycles(8000000);      // Esperar 0.5s 8000000/16000000=0.5
        }

        if(duty1<1000){
            duty1 += 200;          // Incrementar duty cycle en un 20%
            TA1CCR1 = duty1;       // Actualizar el duty cycle
        }
    }

  return 0;
}


void run_ADC(int * temp, int length){
    int i;
    ADC10CTL0 |= ADC10SC;
    while(ADC10CTL1 & ADC10BUSY) /*EMPTY*/; // Esperar al fin de la conversion
    for(i=0; i<length;i++){
        temp[i] = ((BufferADC[i])*500)>>10;
    }
}
```

B.5 Control de refrigeración de los motores

```
#include <msp430.h>
#include "uart_STDIO.h"
#include " init_config .h"
#include <stdio .h>
/*
 * main.c
 */
void run_ADC(int * temp, int length);
int get_Duty(const int tmp);

int BufferADC[4];

int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    int i;

    P2DIR |= BIT0 | BIT2 | BIT3 | BIT4 ; // P2.0, P2.1, P2.3 y P2.4 como salida
    P2SEL |= BIT2 | BIT4; // Asociado al Timer_A1

    P2OUT= BIT0 | BIT3; //Poner P2.0 y P2.3 a HIGH

    conf_reloj (16);          // Configurar reloj DC0 a 16MHz
    init_ADC();               // Iniciar el conversor AD habilitando las entradas de la 6 a la 3
    UARTinit();               // Iniciar el puerto serie 9600 8B N 1bs
                              // P1.1 -> RX
                              // P1.2 -> TX


    int d1 = 0;
    int d2 = 0;

    TA1CCR0 = 1000; // Cargamos el periodo PWM
    TA1CCR1 = d1; // PWM 1 duty cycle
    TA1CCR2 = d2; // PWM 2 duty cycle
    TA1CCTL1 = OUTMOD_7; //Modo7 reset/set
    TA1CCTL2 = OUTMOD_7; //Modo7 reset/set
    TA1CTL = TASSEL_2 + MC_1; // Timer SMCLK Modo UP



    int tmp[4];
    char frase [5];

    while(1){

        run_ADC(tmp,4);

        for(i=0;i<4;i++){
            sprintf ( frase ,"T\%d: \%d",(i+1), tmp[i]);
            UARTprintCR(frase);
        }

        d1 = get_Duty(tmp[0]);
        d2 = get_Duty(tmp[1]);

        sprintf ( frase ,"D1: \%d", d1);
        UARTprintCR(frase);
        sprintf ( frase ,"D2: \%d", d2);
        UARTprintCR(frase);

        UARTprintCR("");


        TA1CCR1 = d1;
        TA1CCR2 = d2;
        __delay_cycles(8000000);                    // Esperar 0.5s 8000000/16000000=0.5
```

```
68        }
69
70      return 0;
71  }
72
73
74  void run_ADC(int * temp, int length){
75        int i;
76        ADC10CTL0 |= ADC10SC;
77        while(ADC10CTL1 & ADC10BUSY) /*EMPTY*/; // Esperar al fin de la conversion
78        for(i=0; i<length;i++){
79                temp[i] = ((BufferADC[i])*500)>>10;
80        }
81  }
82
83  int get_Duty(const int tmp){
84        d=0;
85
86        if(tmp>=90){
87              d=1000;
88        }
89        else if(tmp>=80){
90              d=750;
91        }
92        else if(tmp>=70){
93              d=500;
94        }
95        else if(tmp>=60){
96              d=250;
97        }
98
99        return d;
100 }
```

Índice de Figuras

Índice de Tablas